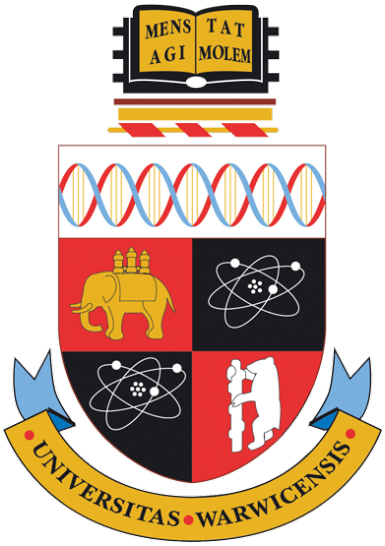
WARWICK
THE UNIVERSITY OF WARWICK



Source Location Privacy-Aware Data Aggregation Scheduling for Wireless Sensor Networks

by

Jack Declan Kirton

Thesis

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

Department of Computer Science

September 2019

Contents

List of Tables

List of Figures

List of Algorithms

Acknowledgements

I am incredibly grateful to my colleagues, family and friends for all of their support during the research conducted and the production of this thesis.

I would like to thank my supervisor, Arshad Jhumka, for his guidance over the past four years. His input has been invaluable and I am truly grateful for his supervision.

I would like to thank my colleague, Matthew Bradbury, for his encouragement during my studies. His insight has always proven to be incredibly useful and I truly value his friendship. Many thanks to Helen McKay for her unwavering support. We met on our first day as undergraduates and worked together through to our respective PhDs. She has been a constant source of inspiration to me and I am fortunate to know her. Furthermore, I would like to thank all of those individuals who have helped me throughout the completion of this thesis.

I would like to thank my parents, Nick and Michele, my sister, Tyanne, and my extended family who have been very supportive these past four years.

Finally, I would like to thank my wife, Brianna, who has stood by me throughout this entire process. She is the light of my life and I could never have done this without her.

Declarations

Parts of this thesis have been previously published by the author in the following:

[82] J. Kirton, M. Bradbury, and A. Jhumka. Source location privacy-aware data aggregation scheduling for wireless sensor networks. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2200–2205, June 2017. doi: 10.1109/ICDCS.2017.171

[83] Jack Kirton, Matthew Bradbury, and Arshad Jhumka. Towards optimal source location privacy-aware tdma schedules in wireless sensor networks. *Computer Networks*, 146:125 – 137, 2018. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2018.09.010. URL `http://www.sciencedirect.com/science/article/pii/S1389128618308958`

Research was performed in collaboration during the development of this thesis, but does not form part of the thesis:

[63] Chen Gu, Matthew Bradbury, Jack Kirton, and Arshad Jhumka. A decision theoretic framework for selecting source location privacy aware routing protocols in wireless sensor networks. *Future Generation Computer Systems*, 87:514 – 526, 2018. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2018.01.046. URL `http://www.sciencedirect.com/science/article/pii/S0167739X17317028`

Abstract

Source Location Privacy (SLP) is an important property for the class of asset monitoring problems in wireless sensor networks (WSNs). SLP aims to prevent an attacker from finding a valuable asset when a WSN node is broadcasting information due to the detection of the asset. Many different methods of protecting the location of a source have been devised for a variety of attacker models. Most common methods of providing SLP operate at the routing level of the network stack, imposing a high message overhead on the SLP-aware routing protocol.

The objective of this thesis is to investigate the novel problem of utilising TDMA slot assignment schedules at the MAC layer in order to provide SLP. These schedules each give rise to different traffic patterns, manipulation of which can be used to divert an attacker away from the asset. Four main contributions are presented. First, a novel formalisation of a parameterised eavesdropping attacker model is created, allowing for comparison of attackers of different strengths. Second, a genetic algorithm is used to generate TDMA Data Aggregation Scheduling (DAS) schedules that contain a diversionary route that leads the attacker away from the source. Third, a distributed algorithm is created to perform the same task while operating online on a WSN. Finally, another distributed algorithm is presented that provides fault-tolerant guarantees with a minimal drop in performance.

Sponsorships and Grants

This research was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) [grant number EP/L016400/1].

Acronyms

CPU Central Processing Unit.

CSMA Carrier-Sense Multiple Access.

DAS Data Aggregation Scheduling.

FT Fault-Tolerant.

GA Genetic Algorithm.

GPS Global Positioning System.

IoT Internet of Things.

IP Internet Protocol.

MAC Medium Access Control.

OS Operating System.

QoS Quality of Service.

RAM Random Access Memory.

RFID Radio-Frequency Identification.

ROM Read-Only Memory.

SLP Source Location Privacy.

TCP Transmission Control Protocol.

TDMA Time Division Multiple Access.

UDP User Datagram Protocol.

WPAN Wireless Personal Area Network.

WSN Wireless Sensor Network.

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) are collections of small computers known as *nodes*. These nodes are most often constrained in their capabilities and resources, such as compute ability, memory systems and power consumption. They are equipped with sensors to allow them to detect properties of their environments, and a radio to allow them to communicate with other nodes. They have only recently become a feasible solution to problems due to the decrease in size of hardware, better battery technology and lower costs for the equipment. As the communication range of a node does not typically encompass the entire network, transmission of information must be done in multiple *hops*. That is, data must traverse multiple wireless links between nodes to reach the data collection point (known as the *sink*).

Due to a node's lack of need for any existing infrastructure (such as power or network cables), they can be deployed in a wide range of environments to tackle a large variety of problems. A range of applications are presented in Subsection 2.1.1. This lack of need for infrastructure makes WSNs very flexible and easy to deploy, which is one of their significant advantages.

WSNs do however suffer from security issues. This is largely because of the transmission of data through the wireless medium, where anyone with sufficient equipment can eavesdrop on the communication due to its broadcast nature. These issues also tend to be more difficult to resolve, in parts due to a WSNs decentralised nature and the limited capabilities of the sensor nodes themselves.

Threats against privacy can be classified into two classes: *content-based* threats and *context-based* threats. Content-based threats are those that typically come to mind when discussing privacy. These threats are those where an attacker will attempt to directly recover data, such as the content of messages. Messages can be protected using encryption to combat this threat. Context-based threats relate to the context in which the data was recovered. For instance, in the case of

messages, this could be information such as broadcast time, message size, message location in the network, etc. Essentially this is inferring information from the context of recovering the data, rather than directly reading any of the data. Context has multiple attributes that encompass the situational aspects of broadcasted messages, including environmental and temporal information. Examples include temporal privacy (keeping the time that events occur private), sink location privacy (hiding the receipt collecting information from the network) and source location privacy (hiding the source of data in the network).

Context-based threats cannot be solved using encryption, so other techniques are necessary.

1.1 Motivation

1.1.1 Source Location Privacy

Source Location Privacy is important in asset monitoring WSNs. Asset monitoring is the use case of sensing specific valuable objects in the environment and tracking their state and location. One example of this is tracking animals in their natural habitats in order to protect them or learn about them. However, if no SLP solution is used, an attacker can use context-based inference to track the source of messages and as such the location of the asset. As the majority of these networks are designed to protect an asset, this is highly undesirable.

To trace messages back to the source, all an attacker needs is a directional antenna to know the direction of where the message is being sent from, and as such can track messages back, node-by-node, to the source. Once the attacker has reached the source node, the asset will likely be nearby and thus easy to capture. More information on SLP is provided in Section 2.2.

1.1.2 Medium Address Control

Medium Access Control (MAC) is a layer of control in the typical network stack, positioned directly above the physical layer. The MAC layer is responsible for controlling access to the physical medium and defining the protocol by which two devices can communicate over it. The protocol defines the method by which data can be transferred between two devices and does not enforce any rules on what that data is or how it should be interpreted by higher levels in the network stack.

The majority of protocols providing SLP are positioned at the routing layer of the network stack. While this provides more information than that available at the MAC layer, routing layer solutions typically send more control messages to establish

paths through the network and to provide nodes with special instructions. At the MAC layer, the routing of such control messages over multiple hops is not possible. The motivation of this is that a MAC layer solution could be utilised as another or alternative layer of protection.

1.1.3 TDMA DAS Convergecast

Time Division Multiple Access (TDMA) is a method of dividing the physical medium to provide every device with a time slot in which to broadcast, eliminating the problem of packet collisions within those slots. Time slots are negotiated and assigned in the beacon phase, followed by the individual time slots. The beacon phase and time slots together are known as a TDMA period. TDMA slot assignment occurs at the MAC layer of the network stack.

Data Aggregation Scheduling (DAS) Convergecast is the process of having a "parent" node collect data from each "child" before aggregating that data and then sending it on to the next parent. The final recipient of the data aggregated from the whole network is a dedicated data collection node. The motivation behind using a convergecast method is that the aggregation of the application-specific data stops the necessity of every node forwarding data to the data-collection node, which reduces network traffic and as such saves energy.

TDMA DAS is a system of assigning node slots in a decreasing fashion starting from the data collection node and spreading outwards to the edges of the network. This guarantees that any parent node transmits in a later slot than its children, and as such a message sent from any node in the network will reach the data-collection node within a single TDMA period in an aggregated form. This provides an upper bound on the latency, which can be an issue in networks where TDMA slots are assigned in other ways. The motivation for using TDMA DAS is that it largely operates at the MAC layer and as such is a suitable target for modification to provide SLP. This is discussed further in the next section.

1.1.4 Source Location Privacy using TDMA DAS

The various SLP solutions mostly focus on modifying the attacker movement patterns at the routing layer, compared to some baseline protectionless routing scheme (such as flooding), such that the attacker takes a longer time to reach the source node. Different SLP techniques produce different attacker movement patterns based on how they provide SLP. Phantom routing (shown in Figure 1.1a) tends to produce a zig-zag pattern as the attacker moves back and forth between phantom nodes. If phantom nodes tended to be allocated in the same area the attacker would double

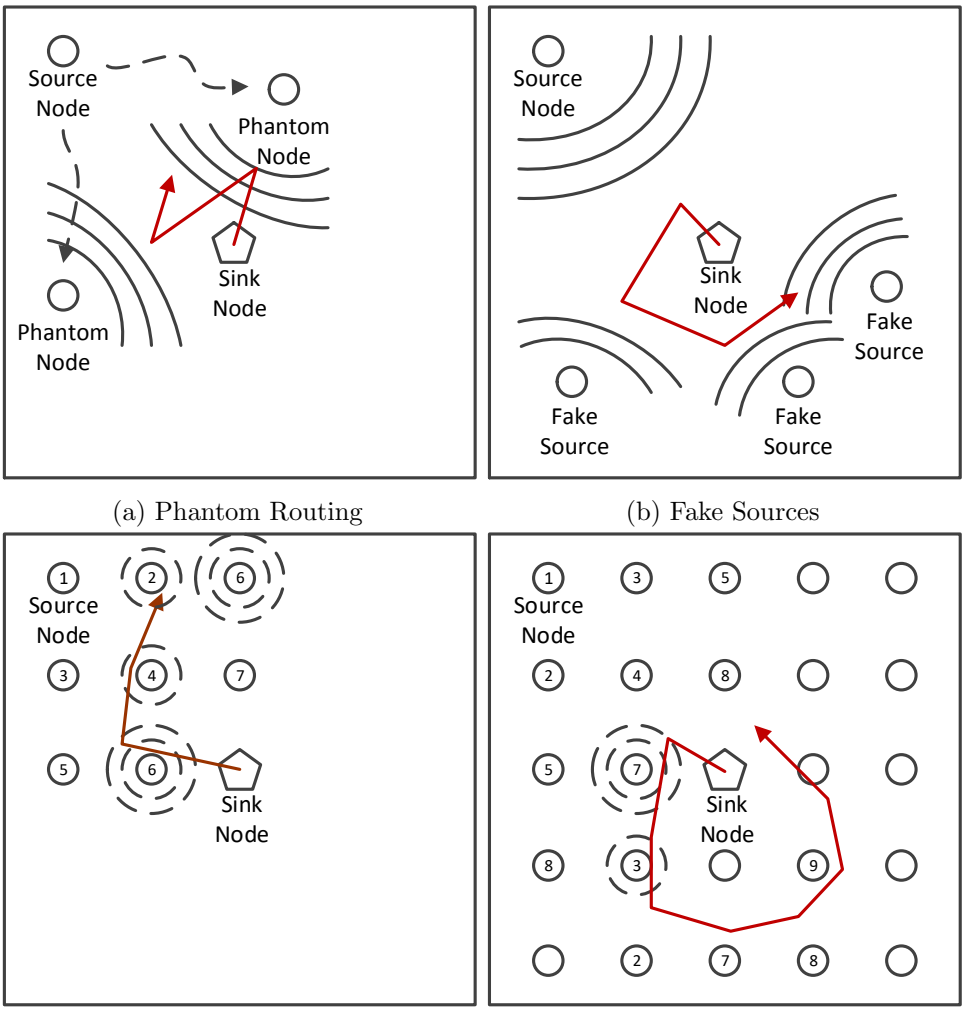
back on itself less. On the other hand, fake sources (Figure 1.1b) offer a different type of movement pattern, as the fake sources are not connected to the real source by a message path. Rather than zig-zagging between phantom nodes, the attacker tends to zig-zag between the source and the fake sources. The parameters are typically set such that the fake sources have a greater pull so the attacker's overall movement is towards the fake sources rather than towards the real source. The main disadvantage of routing layer techniques is the associated message overhead that is required to cause the attacker to zig-zag through the network.

The movement pattern for TDMA convergecast (without SLP) is typically that of a protectionless scheme, with the attacker moving towards the real source each time it receives a new message (see Figure 1.1c). The attacker may even have an additional advantage when TDMA is used because the slots are set up such that it will always travel along the shortest path to the source. However, the zig-zag movement of an attacker, previously seen when routing layer techniques are used, can be achieved at the MAC level by a random assignment of slots to nodes, i.e., the slot assignment is not properly convergecast. The disadvantage of this is the delivery latency of messages but the main gain is that the attacker zig-zags through the network at *no additional* message overhead.

To achieve a trade-off between good message delivery latency and good SLP with no message overhead, TDMA DAS scheduling is proposed (see Figure 1.1d) that provides SLP with the aim of luring the attacker away from the source by taking it on a *diversionary* route through the network, thus preventing the attacker from following the shortest path to the source. By appropriately modifying the slot assignment, the diversionary path can be ensured as the next node in the path will be set to broadcast earlier than any of the other neighbouring nodes. This diversionary route increases the time the attacker would need to capture the source.

1.1.5 Fault Tolerance

Fault tolerance is an important consideration for any application of WSNs. WSNs are prone to faults, either from noisy environments for wireless communication through to more serious node crashes which can occur through hardware defects, buggy software, power issues (such as an empty battery), physical manipulation by an adversary and many more. It would be foolish to employ a WSN lacking some form of fault tolerance with so many potential risks that exist in real-world deployments. Therefore, fault tolerance is a motivating factor to ensure applicability to real-world scenarios.



(a) Phantom Routing

(b) Fake Sources

(c) TDMA DAS

(d) TDMA DAS with SLP

Figure 1.1: SLP technique attacker movement patterns

1.2 Contributions

The contributions in this thesis present **methods of providing source location privacy from the MAC-layer only in a TDMA DAS setup**. The focus is to explore MAC-based methods rather than the more common routing-based counterparts in order to provide **another or alternative layer of protection for minimal overhead**. The following contributions are made:

- Chapter 3 presents a new parametrised attacker model capable of representing a range of easily comparable attackers, including the one typically used in SLP research. Also presented is an algorithm that checks if a TDMA schedule is SLP-aware, returning a counterexample if not, similar to model checking.
- Chapter 5 presents a near-optimal method of producing SLP-aware TDMA DAS schedules offline using a Genetic Algorithm (GA). Suitable genetic operators are designed to create only valid DAS and SLP-aware schedules. Two different fitness functions are used, one reducing slot utilisation for lower latencies and one leading the attacker as far from the source as possible. Pareto efficiency is examined for the solutions between the two fitness functions. Many of the generated schedules are then implemented and simulated to ensure correctness and near-optimal capture ratios.
- Chapter 6 presents an online, distributed algorithm that provides some SLP-awareness, but not optimal due to a lack of knowledge of where the source is located. A number of impossibility results are proven and the three-stage algorithm is presented. This algorithm is implemented and simulated to show the difference compared to a protectionless TDMA DAS, which shows a reduction in capture ratio.
- Chapter 7 then proceeds to show a fault-tolerant version of the distributed algorithm, containing five stages. The complexity of fault-tolerant DAS is proven and the five-stage algorithm is presented. This is then implemented and simulated to show that Fault-Tolerant (FT) SLP DAS has comparable performance to SLP DAS even in the presence of crashes.

1.3 Organisation

This chapter introduced SLP in WSNs and stated the contributions this thesis makes. The remainder of the work is as follows:

- Chapter 2 presents the background of WSNs and SLP.

- Chapter 3 introduces the models and specifications used for the remainder of the thesis.
- Chapter 4 explains the setup and parameters for the simulations executed as results throughout this thesis.
- Chapters 5 to 7 present the main contributions of the thesis.
- Chapter 8 discusses common patterns that have arisen from this research, concludes and presents potential future work for continuation of the research.

Chapter 2

Background

In this chapter, WSNs will be described, including their limited nature and the challenges they face. The concept of Source Location Privacy (SLP) will be introduced followed by work in the research area.

2.1 Wireless Sensor Networks

2.1.1 Applications

WSNs have a wide variety of applications. A handful shall be presented in this section.

Monitoring the health of structures [16] is very important to ensure people's safety. Such applications include monitoring heritage buildings [28] and suspension bridges [30].

Another common use case is monitoring habitats and wildlife [19, 29, 54]. This can be to protect endangered species, recover animals or just monitor for scientific purposes.

Another use is during natural disasters, such as predicting [69] or providing information during [67] forest fires. This can provide very valuable detail on the spread of forest fires and where resources should be deployed.

Other uses include target tracking and classification [8], detection of atmospheric conditions such as air pollution [58], industrial applications [64] and a significant amount more [7, 10].

2.1.2 Challenges

Programming

Programming of sensor nodes is significantly different than that of typical programming efforts and this is largely due to a sensor node's resource-constrained nature. Any implementation of the desired behaviour for the node must comply with incredibly limited compute power, very small memory allocation of both runtime allocations and the static size of the binary and also the total power consumption of the node. This is not to say applications involving more performant devices (such as smartphones, personal computers and servers) aren't bound by resource usage requirements, but the problem becomes significantly more apparent with sensor nodes, as they are several orders of magnitude less capable than these other devices (especially in terms of available power).

While much of software engineering has become more abstracted from the specific of the hardware (by using high-level languages, etc.), implementations for sensor nodes require programming at a low-level, which is generally considered to be more difficult with a steeper learning curve and being less accessible than higher-level methods.

Due to these issues, OSes specifically designed for the nature of these devices have been created, such as TinyOS [85] and Contiki [45]. Each have their pros and cons, but still require low-level programming, careful consideration of resource limitations and do not provide many of the core features that are expected of modern OSes (TinyOS does not even support dynamic memory management!). Many attempts have been made to simplify this programming model. For instance, EnviroTrack is a middleware layer that raises the programming abstraction level [1]. People have attempted to make the process more modular [55], adaptive [51], application-specific [70] and even go as far as to design a higher-level scripting language for these devices [47]. However, programming sensor nodes still remains a fairly low-level and specialised task.

Communication/Networking

Communication between nodes is a fundamental property of WSNs. In the early stages of WSNs, custom protocols were used between different hardware devices and also in the various OSes. This is because Internet Protocol (IP), the de-facto standard for the majority of computer communication, in both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) formats, is a fairly data-intensive protocol. It uses large headers in each packet which are often unnecessary in WSNs. However, with the rise of Internet of Things (IoT), it is becoming more desirable for

sensor nodes to be able to communicate with the wider internet ecosystem. This is why considerable effort has been made to bring TCP/IP to these devices [48, 76], including making them IPv6 ready [53] and even bringing the TCP/IP stack to 8-bit processors [46].

Energy

One of the biggest problems that sensor nodes face is energy. Due to their lack of connection to any existing infrastructure, they rely on batteries and possibly power they can gather from the environment (e.g. solar). There are many design challenges to creating sensor networks because of energy constraints [61]. Processes as simple as turning off the radio when it is not needed can save huge amounts of power, as the radio is typically the most energy-hungry component of the device. There exist many energy-saving technologies [66], but developing a WSN application still requires intelligent design decisions to get the largest lifespan from the nodes as possible. Online energy usage estimations [52] can be used to determine an appropriate course of action, such as changing the routing system so that nodes low on power are sending less messages [40].

Scalability

Scalability is the problem WSN applications face when attempting to escalate the size of the solution. Typically, this is the process of deploying an application in a larger environment using an increased number of nodes and, as such, transmitting and processing more data. Scalability is a challenge for the future of WSNs [56]. As WSNs and IoT come to the forefront of technology, larger and larger use cases are being developed. Ensuring that the system is scalable is key to future-proofing the application.

This can be a particularly important property for a lot of SLP use cases. Monitoring assets over large areas using significant numbers of sensor nodes would be a reasonable application of the technology. For example, monitoring endangered wildlife over their natural habitat in order to learn about and protect them would likely involve a large physical area requiring many nodes.

Reliability

There are a number of reliability issues that plague WSNs. The first is link failures. Link failures occur due to radio interference between a pair of communicating nodes, which prevents the communication. This is an incredibly common occurrence in a real-world deployment. Technologies are used in an attempt to work around this

issue, such as link quality estimation [13]. The second issue is node crashes. This can be due to incorrect software, damaged hardware, empty batteries or any number of other reasons. In any case, fault-tolerance is an important design feature for any WSN application [73]. Detection and repair of software errors online [71] has even been considered a possibility.

Reprogramming

Reprogramming a WSN after deployment is often a difficult task. Firstly, the original creators of the WSN must have provided a method to update firmware over the network, unless reprogramming nodes one-by-one is an option although this is usually not feasible. Then, the security of code distribution must be considered [39]. Without any security, any person could change the WSN's programming, which is obviously a serious flaw. Thirdly, the method of reprogramming must be considered. Typically, a node must have new firmware "flashed" to its internal Read-Only Memory (ROM). However, other methods such as runtime dynamic linking [49] are now a possibility, depending on node hardware.

Security

Finally, one of the largest issues in WSNs is security [11, 98]. Specific attacks include the denial of sleep attack [26], which keeps a node awake with the radio on in an attempt to waste energy and reduce the lifespan of the device. Taking control of nodes in the network [81] is another example.

Much work has been performed in an attempt to combat security threats in WSNs [33]. New, more secure protocols have been created [97]. Methods of protecting network traffic have been invented [74]. A large amount of this has focused on the detection of code bugs (where the majority of security defects are introduced). Static analysis [57, 110] is common, used in an attempt to catch security flaws before the code has been released. Flaws in code are not the only way a security issue arises, there can often be problems with the hardware itself [105].

Encryption is often a first response to security. The theory behind this entire thesis is that encryption is not sufficient to protect a WSN completely (or for that matter, most systems), although it is still a very important tool. Cryptography becomes even more difficult when dealing with such constrained devices as sensor nodes [88, 100].

The reason that encryption is not sufficient for context-based attacks is that these attacks do not rely on the data being protected by the encryption mechanism. For instance, information that can be used by such attacks includes the time that the message was sent, the size of the message, time between the sending of messages

and any observable environmental stimuli that may be influencing the system. None of these properties can be encrypted as they are situational data observed by the attacker, not anything stored in the message itself. This information can then be used to infer further details about the system. At no point is any data contained within the message used, rendering encryption ineffective against this type of attack.

Throughout the remainder of this thesis, an assumption is made that each message is encrypted sufficiently to prevent an attacker from being able to read its contents in a reasonable amount of time. This makes context-based attacks a more attractive solution.

2.1.3 Node Composition

Nodes are comprised of a low-power Central Processing Unit (CPU), some small amount of Random Access Memory (RAM), some small amount of ROM, a wireless radio, some power source (most commonly a battery) and any sensors required for the application. There are a number of reasons for such meagre specifications in nodes, but the most important being to keep costs low, as WSNs can be used to span large areas of the environment with many nodes participating in the network for a feasible cost.

2.1.4 Sensors

The sensors given to a node can be used to sense anything about the node's environment (or itself), but some of the most common criteria to sense in the environment are:

- Temperature
- Light
- Humidity
- Distance
- Motion
- Radio-Frequency Identification (RFID) chips
- Location

2.1.5 Network

Routing

There are many different methods of routing messages in WSNs [3, 4]. The simplest method is flooding [68], where when a node hears a message, it re-transmits it. This obviously has flaws such as large amounts of network traffic and as such high energy consumption. Different routing protocols have been created to solve different problems, such as routing with guaranteed delivery [20], routing to achieve the maximum lifetime of the network [32] and even protocols that contain a mix of goals, such as [2] with the goal of energy-awareness and Quality of Service (QoS) and [12], with the goal of being secure and efficient. Some work has tried to solve more specific problems, such as routing in a network containing multiple sinks [35].

MAC

Routing typically receives more attention than the MAC layer. MAC protocols tend to focus on reducing energy consumption by duty cycling, i.e. turning the radio off when possible [14, 18, 27, 75]. This thesis focuses on the use of a TDMA [9] MAC protocol to provide SLP with minimal message overhead.

2.2 Source Location Privacy

The SLP problem was first introduced in [80, 95] as the panda-hunter game, in which a WSN had been deployed to monitor valuable assets where the messages are routed from the node that detects the asset (*source node*) to a base station (*sink node*). There have been many techniques proposed to provide SLP [36, 99]. In the panda-hunter game there is an attacker who is attempting to locate the asset by tracing back the wireless messages being broadcasted by the sensor node that detected the asset. The authors proposed *phantom routing*, a two stage protocol, where the message generated by the source firstly takes a directed random walk through the network to a *phantom node* and then secondly this message is forwarded to the sink. Two variants of phantom routing were proposed, PRS [95] that flooded the message from the phantom node to the sink and PSRS [80] that used single-path routing instead of flooding.

Since phantom routing's introduction many other pieces of work have focused on improving the directed random walk stage. GROW [112] introduced storing nodes traveled along the directed walk into a bloom filter to prevent the directed walk turning back on itself. A similar outcome was achieved by [111] which used location angles to direct the random walk away from the source. A technique similar

to phantom routing and its extensions are the ring techniques [93, 113] where rings are formed around the sink and generated messages are first forwarded through the rings before being routed to the source.

However, there have been many works highlighting the deficiencies of phantom routing. Such as lacking suitable neighbours for the path to continue [114] and performance degrading when multiple sources are present [62]. In terms of new types of attack, the random walk has also been shown to reveal information about the location of the source [104] where a traffic-analysis attack developed based on random walks hitting the network boundary allows prediction of the source's location. Finally, an alternate criticism is that many existing solutions focusing on just providing SLP, [101] introduces an algorithm that provides identity, route and location privacy.

Many other routing techniques have also been used, one popular technique is the use of *fake sources* that send message encrypted and padded to be indistinguishable from the real messages generated by the source. The aim of fake messages is to lure the attacker to the fake source rather than the real source. A fake source technique was originally proposed in [80] but the class of techniques was discounted due to the poor performance. [77] implemented an alternative fake source technique showing high levels of source location privacy are possible with fake sources. This style of fake sources was extended in [23, 79] to widen the applicability of the solution and allow it to dynamically determine parameters online.

Other than pure fake source techniques, there exist many that combine fake sources with random walks. One such technique is [89] that created a routing tree where leaf nodes would send fake messages up the tree. Another is fog routing [43] whereby the network is separated into dynamic *fogs*, which involves creating routing loops in order to force the attacker into a continuous, inescapable path. The fogs also create fake messages inside themselves. PEM [107] also creates fake sources during network execution which perform a walk about from the sink while continuing to send fake messages. A downside to fake message based solutions is that they tend to have a much higher energy consumption compared to routing-based techniques.

There have been fewer techniques that use a cross-layer approach to providing SLP. In [103] beacon frames used by the MAC layer are modified to help propagate messages away from the source before standard routing is used to send the message to the sink. A second technique was also proposed that routed the message to a pivot node after the first round of beaconing which would then perform a second round of beaconing before routing the message to the sink.

So far all the solutions presented focus on a local distributed eavesdropper [15]. There are many other types of attackers that SLP could be provided against. For example, global attackers that can view the entire network will be able to defeat

solutions that target local attackers immediately after a source sends its first message. Two solutions that provide perfect privacy against global attackers are Periodic Collection [90] which has every node periodically broadcast a message even if there is no message from the source to send and ProbRate/FitProbRate [102] which also broadcast periodically, but use an exponential distribution to determine the period between broadcasts.

At the time of writing, there have been no proposals of an SLP solution that operates purely on the MAC layer. This thesis advances the state-of-the-art by proposing such solutions.

Section 1.2 outlines the major contributions of this thesis. This section provides explanation of how each of them advances the field:

1. The first major contribution is the creation of a formalised attacker model for use in SLP research. The purpose of the model is to simply create sets of distributed eavesdropping attackers with varying capabilities by only adjusting the model parameters. This allows testing of algorithms and solutions against this range of attackers to determine the effectiveness of the solution against attackers of varying abilities, while providing easy comparison between the attackers. It also provides a common base for comparing different solutions.

2. The second major contribution is the presentation of an offline TDMA DAS schedule generator that provides near-optimal SLP. The generator uses a GA to refine the schedule over a number of generations. The schedules created are implemented and simulated to show that they provide near-optimal capture ratio. Some work has been performed around generating TDMA schedules using genetic algorithms (as described in Chapter 5), but no work has been done with this technique to create DAS-compliant schedules or in the area of SLP, so combining these elements is indeed novel.

3. The third major contribution is the design and implementation of an online SLP-aware TDMA DAS algorithm using the same basic principles as the offline schedule generator. Lacking a global view of the network, not having source location information available and having all computing be performed in a distributed fashion on individual nodes while the network is in operation requires a significantly different design to the offline generation. The significance of this contribution is showing the possibility of implementing a purely MAC-layer protocol that increases the level of SLP that the network provides.

4. The final major contribution is the adaption of the SLP TDMA DAS algorithm to provide guarantees of fault tolerance during operation. By dynamically

altering the schedule, crashed nodes can be avoided. This improvement is simulated and shows comparable performance to the non-fault-tolerant algorithm, even in the presence of crashes. Minimal consideration has been given to fault tolerance in SLP solutions, even though fault tolerance is a key component in any real-world application. Providing an improved algorithm with fault-tolerant guarantees is a further step towards showing that these methods can be used effectively in real-world deployments.

Chapter 3

Models & Specifications

This chapter defines models and specifications to be used throughout the remainder of this thesis. Two contributions are made. Firstly, the attacker model is a new and novel method of creating comparable attackers whilst still able to represent the typical attack used in SLP. Secondly, an algorithm for checking SLP-awareness in TDMA schedules is created, returning a counterexample if it is not, similar to model checking.

3.1 Distributed Algorithm Model

3.1.1 Topology and Processes

A WSN node is a computing device associated to a unique identifier. Communication in WSNs is typically modelled with a circular communication range centered on a node, and assuming all nodes have the same communication range. With this model, a node is thought as able to exchange data with all devices within its communication range. In graph-theoretic terms, a WSN is represented as an *undirected* graph $G = (V, E)$ with a set V of vertices representing the nodes, and a set E of edges representing the communication links between pairs of nodes.

The actions taken by the individual nodes are dictated by the *process* running on it. The system thus consists of a finite set Π of $\gamma > 0$ processes $p_1 \dots p_\gamma$, where each node in the network runs a process[1]. Adjacent processes, defined by the physical topology, are linked by wireless channels, represented as edges. Each process contains a non-empty set of variables and actions, also called steps, depending on an algorithm $\mathcal{P}$. A variable v of process p is denoted by $p.v$. An assignment of values to variables in a program is called a *state*.

[1]The terms node and process are used interchangeably where no ambiguity can arise.

3.1.2 Program Syntax and Sematics

This section defines the syntax used in programs representing a distributed algorithm for a set of nodes, or an attacker model. Programs are written in the guarded command notation (GCN) [41]. Hence, an action has the form $\langle name \rangle :: \langle guard \rangle \rightarrow \langle command \rangle$. In general, a *guard* is a predicate defined over the set of a process' variables. When a *guard* evaluates to true, the *command* can be executed, which takes the program from one state to another. When the state transition is complete, the event $\langle name \rangle$ has occurred. A *command* is a sequence of assignments and branching statements. A guard or command can contain universal or existential quantifiers of the form: $\langle quantifier \rangle \langle boundvariables \rangle : \langle range \rangle : \langle term \rangle$, where *range* and *term* are Boolean constructs. When a guard evaluates to true in a state, the corresponding action is *enabled* in that state. A special **timeout**(*timer*) guard evaluates to true when a *timer* variable reaches zero. A **set**(*timer*, *value*) command sets the timer variable to a specified value.

This notation has been chosen for several reasons. First, it is usually simpler to formally reason on a program execution in terms of what guards become true at a given point in the execution, rather than following a specific control flow. Also, GCN matches the event-driven programming style of many WSN software platforms [72], where the binding of events to their handlers is, in a sense, corresponding to the evaluation of guards.

The execution of a step of an algorithm $\mathcal{P}$ causes the process to update one or more variables and moves the system from one state to another in one atomic step. In a given state s, several processes may be enabled, and a decision is needed about which one(s) to execute. The subset of processes that take a step when possible is chosen according to different scheduling policies. To ensure the system makes progress, a notion of fairness is also required. Whenever any of the enabled processes can take a step independent of all others, that is, a continuously enabled action is eventually executed, then the system is weakly fair [42] and runs in an *asynchronous* manner. This entails there is *no bound* on relative process speeds and message transmission time, which is assumed here for the slot assignment problem addressed.

3.1.3 Communication

Each process has a special *channel* variable, denoted by *ch*, modelling a FIFO queue of incoming messages sent by other processes. This variable is defined over the set of (possibly infinite) message sequences. An action with a **rcv**(*sender*, *msg*) guard is enabled at process j when there is a message at the head of the channel variable *sender*.*j*.*ch* of a process. Executing the corresponding action causes the message at

the head of the channel to be dequeued, while *msg* and *sender* are bound to the content of the message and the sender identifier.

Differently, the **send**(*msg*,*dest*) command causes message *msg* to be attached to the tail of the channel variable *ch* of processes in the *dest* set. To capture the broadcast multi-hop nature of WSNs, the semantics of **send** when executed on node n depends on the processes in *dest*:

- if all nodes in *dest* are in the physical neighbourhood N of node n, that is, $\forall i \in dest : i \in N$, then *msg* is simultaneously appended to the tail of the channel variable at all processes in *dest*;
- if *dest* is a predefined value $BCAST$, then message *msg* is simultaneously appended to the tail of the channel variable *ch* of all processes that are in n's *physical* neighbourhood: this implies the message reaches processes that may not appear in the sender's logical states, modelling the semantics of physical broadcast in WSNs;
- if at least one node in *dest* is not in N, then message *msg* is appended to the tail of the channel variable *ch* at all processes in *dest* possibly at different times, modelling multi-hop transmissions.

3.1.4 Node Crashes

Node crashes are only investigated in Chapter 7. One type of fault is considered, node crashes, where there are up to f node crash faults. This could be due to nodes running out of energy or being physically damaged.A fault is modelled as a fault action, i.e., a fault action has been executed when a fault occurs. A fault action is identical to a normal program action in that it changes the state of program by changing the values of variables. Since a node crash can occur at any time, the fault action is considered to be *continuously enabled*. However, unlike program actions, fault actions are not required to execute even when continuously enabled, i.e. only weak p-fairness is considered (the weak fairness assumptions only applies to actions of program p and not to fault actions). An assumption is made that the sink does not crash.

3.2 Attacker Model

An attacker $\mathcal{A}$ is a distinct process that is characterised by its presence and the attacks (or actions) it can carry out [15]. It is assumed that the attacker to be a *distributed eavesdropper*, i.e., the attacker will be located at various positions in

the network (at possibly different times, hence distributed) and will only listen to messages being transmitted (eavesdropper). Based on both attributes, the attacker process $\mathcal{A}$ will update its state.

This thesis proposes a novel and generic model of a distributed eavesdropper, as shown in Figure 3.1. The attacker, called a (R, H, M, s_0, D)-$\mathcal{A}$ attacker, is parametrised as follows: (i) R is the number of messages the attacker can receive before it makes a move, (ii) H is the number of the most recently visited locations it can record, (iii) M is the number of moves the attacker can make in a given period, (iv) s_0 is the starting point of the attacker, and (v) D is a function for the attacker that returns the set of possible new locations, based on the current data (e.g, messages heard) and historical data (e.g., previously visited locations). This parametrised attacker allows the development and understanding of attackers of various strengths.

Most work on SLP has focused on the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, where D is defined as follows: since $R = 1$, $M = 1$ and $H = 0$, it means that, when the attacker hears the first message coming from a location j, it will move to j. s_0 is typically the sink, and will be defined as such in this work going forward.

The necessity for the creation of this model is to provide a simple method of generating attackers of different capabilities and overall power in order to easily compare either varying attackers on a single solution or using a single attacker to compare various solutions. The formalised definition with a small set of parameters aids this comparison and provides detail on which attributes of the model make the largest difference. Due to the attacker model having a single algorithm tuned with multiple parameters, it provides a simpler method of testing a variety of attackers against a solution than designing/implementing several algorithms for the different capabilities that need to be tested. An important aspect of the model is that it is also capable of representing the standard attacker model used in many SLP works, allowing that instantiation of the model to compare with existing work.

3.3 Specifications

This section details the specifications of the various components of the problem.

3.3.1 Data Aggregation Scheduling

In this section, DAS is defined in three increasingly weaker variants.

Definition 1 (Non-colliding slot)**.** *Given a network $G = (V, E)$, a node $n \in G$, and a slot i in which n can transmit its payload, i is non-colliding for n iff $\forall m \in$*

Attacker Process (R, H, M, s_0, D)-$\mathcal{A}$ where R is the number of messages an attacker can capture before making a move, H is the number of the most recently visited locations, M is the number of moves the attacker can make in a single period, s_0 is the initial location of the attacker $\mathcal{A}$, and D is a function that $\mathcal{A}$ uses to select its next location, based on its history and set of messages captured.

variables
% history of recent locations, current location
history[], curLoc : array of node ids of size H, int **init** $\langle\rangle, s_0$;

% set of messages received, number of moves made, history index
msgs, moves, num: set of msgs, int, int **init** $\emptyset, 0, 0$

constants
period: timer; % obtained from DAS schedule denoted by α

actions
NextP:: **timeout**(*period*) $\rightarrow$ *msgs, moves* $:= \emptyset, 0$;
set (*period*, α)
ARcv:: **rcv** $\langle l, m\rangle \rightarrow$
if $|msgs| < R$ **then**
msgs $\cup \{l\}$;
fi;

Decide:: *msgs* $\neq \emptyset \rightarrow$
if (*moves* $<$ M) **then**
if $(H > 0)$ **then**
history[num] := *curloc*;
num := (*num* + 1) mod H;
fi;
curloc := D(*msgs*, *history*);
moves, msgs := *moves* + 1, $\emptyset$;
fi

Figure 3.1: A (R, H, M, s_0, D)-attacker

$CG(n) \cdot m.slot \neq i$, with $CG(n)$ being the set of nodes in the 2-hop neighbourhood of n.

Definition 2 (Strong DAS)**.** *Given a network $G = (V, E)$, a strong TDMA-based collision-free data aggregation schedule is a sequence of sets of senders $\langle \sigma_1, \sigma_2, \ldots, \sigma_l \rangle$, satisfying the following constraints:*

1. $\sigma_i \cap \sigma_j = \emptyset, \forall i \neq j$,
2. $\bigcup_{i=1}^{l} \sigma_i = V \setminus \{S\}$,
3. $\forall n \in \sigma_i, 1 \leq i \leq l-1 : \forall m \in N, n \cdot m \ldots S$ *is a shortest path* $: (m \in \sigma_j, j > i) \vee (m = S)$
4. $\forall m, n \in \sigma_j, 1 \leq j \leq l-1 : m \neq n \Rightarrow n \notin CG(m) \wedge m \notin CG(n)$.

There are four conditions that capture the strong data aggregation scheduling problem:

1. The first condition stipulates that nodes are allocated at most one time slot in the schedule.
2. The second condition implies that all nodes (apart from the sink) will get at least one transmission slot. Taken together, conditions 1 and 2 state that every node (except the sink) will transmit in exactly one slot in the schedule.
3. The third condition captures the fact that whenever a node transmits a message, all of its neighbours that are closer to the sink will transmit in a later slot, and the condition captures the notion of strongly proper slots.
4. The last condition captures the condition for collision freedom, i.e., two nodes can transmit in the same slot only if they are not in each other's collision group.

Definition 3 (Weak DAS)**.** *Given a network $G = (V, E)$, a weak TDMA-based collision-free data aggregation schedule is a sequence of sets of senders $\langle \sigma_1, \sigma_2, \ldots, \sigma_l \rangle$, satisfying the following constraints:*

1. $\sigma_i \cap \sigma_j = \emptyset, \forall i \neq j$,
2. $\bigcup_{i=1}^{l} \sigma_i = V \setminus \{S\}$,
3. $\forall n \in \sigma_i, 1 \leq i \leq l-1 : \exists m \in N, n \cdot m \ldots S$ *is a shortest path* $: (m \in \sigma_j, j > i) \vee (m = S)$
4. $\forall m, n \in \sigma_j, 1 \leq j \leq l-1 : m \neq n \Rightarrow n \notin CG(m) \wedge m \notin CG(n)$.

The main difference between the strong and weak versions of DAS is that the strong version requires a node to have all of its neighbours closer to the sink with a slot higher than its own, i.e., for a given node n, *all* of its neighbours that are closer to the sink need to transmit at a later time than itself. On the other hand, the weak version requires at least *one* such neighbour to transmit later.

Definition 4 (Convergecast)**.** *Given (i) a network $G = (V, E)$, (ii) a children function $\mathbb{C}$, (iii) a sink-distance function δ, and (iv) a schedule Ξ, Ξ is called a* convergecast *for G iff:*

1. *Same as Definition 2, item 1*
2. *Same as Definition 3, item 2*
3. $\forall m \in V : \forall n \in \mathbb{C}(m) \cdot \Xi(m) > \Xi(n)$
4. *Same as Definition 2, item 4*

With 3 stating every node has a lower slot than its parent.

It has previously been shown that there exists no solution for strong DAS [78], the specification of which has been provided for completeness. On the other hand, protocols exist for both weak DAS and convergecast.

3.3.2 Safety Period

The notion of SLP awareness in TDMA-based DAS scheduling is now introduced. To do this, a concept called *safety period* is required, which was introduced in [80] to determine the number of messages needed to capture a source. As provided in [24], an alternative but analogous definition of safety period is being used where the notion of time-to-capture is used over messages-to-capture. This is because the metric of counting messages until capture can potentially be very large or even unbounded (when the attacker never captures the source), where employing a time limit prevents such behaviour. The rationale for this is to enforce a finite runtime to create results, such as capture ratio, experimentally using simulations or otherwise.

Definition 5 (Capture Time of $\mathcal{P}$)**.** *Given network $G = (V, E)$, an algorithm $\mathcal{P}$, a (R, H, M, s_0, D)-$\mathcal{A}$ attacker, a source $\mathcal{S}$, the capture time $\delta^G_{\mathcal{P},\mathcal{A}}$ of $\mathcal{P}$ in the presence of $\mathcal{A}$ in G is the* minimum *time taken for $\mathcal{A}$ to capture $\mathcal{S}$ in G, i.e., starting from s_0, $\mathcal{P}$ provides $\delta^G_{\mathcal{P},\mathcal{A}}$-SLP for $\mathcal{S}$ in G in the presence of $\mathcal{A}$.*

Now, if $\mathcal{P}$ is to be transformed into a SLP-aware protocol $\mathcal{P}^s$, there are two options: (i) determine the capture time of $\mathcal{P}^s$ or (ii) determine the probability of

capturing $\mathcal{S}$ by A in G through $\mathcal{P}^s$ during a given time window. In this work, focus is given to the second option. Now, if the time window is less than or equal to $\delta^G_{\mathcal{P},\mathcal{A}}$, then the probability is 0 as $\mathcal{P}^s$ will (intuitively) be expected to have a higher capture time than $\mathcal{P}$. Thus, evaluation of the capture probability of $\mathcal{S}$ under $\mathcal{P}^s$ is needed during a time window larger than $\delta^G_{\mathcal{P},\mathcal{A}}$, which is called the *safety period*.

Then, the *safety period* of $\mathcal{S}$ in G in the presence of $\mathcal{A}$ is defined as follows:

$$\delta^G_{\mathcal{S},\mathcal{A}} = C_s \times \delta^G_{\mathcal{P},\mathcal{A}} \text{ where typically } 1 < C_s < 2 \tag{3.1}$$

Now, if $\delta^G_{\mathcal{P}^s,\mathcal{A}} \leq \delta^G_{\mathcal{S},\mathcal{A}}$ for a given $G, \mathcal{A}, \mathcal{S}$, then the probability is 1.

3.3.3 SLP-Aware DAS

This section presents a decision procedure, called *VerifySchedule*, that verifies whether a given DAS schedule can lead to an attacker capturing the source, i.e., reaching the source within the safety period. The decision procedure, shown in Algorithm 1, works in a way analogous to model checking, where it returns a violating trace to show how an attacker can capture a source.

The decision procedure *VerifySchedule* works as follows: the function GenerateAllAttackerTraces generates all possible traces (i.e., sequence of locations) that the attacker can take. Any such sequence $\langle s_0 s_1 \dots s_j \rangle$ has the property that $\forall\ 0 \leq i < j, (s_i, s_{i+1}) \in E$, i.e., the attacker moves one hop at a time. The trace is then analysed by checking the validity of each step. Each step is checked to see whether this step of $\mathcal{A}$ is allowed by $\mathcal{F}$ (by computing the B variable - Line 8) and by $\mathcal{A}$'s parameters. If a step is not valid, then the trace is discarded. Else, the next step is considered until the source is reached. When the source is reached, the time taken is noted. If it is less than the safety period, there is a capture else subsequent steps are considered. The function returns a boolean value *False* if a trace of $\mathcal{A}$ can be constructed that satisfies both $\mathcal{F}$ and its parameters but ends up capturing $\mathcal{S}$.

Definition 6 (Strong (resp. weak) SLP-aware DAS)**.** *Given a network $G = (V, E)$, a DAS assignment $\mathcal{F}^s$, a (R, H, M, s_0, D)-$\mathcal{A}$ attacker, a given source node $\mathcal{S}$, $\mathcal{F}^s$ is a strong (resp. weak) SLP-aware DAS protocol for $\mathcal{S}$ in G in the presence of $\mathcal{A}$ if and only if:*

1. *$\mathcal{F}^s$ is a strong (resp. weak) DAS.*
2. *$\delta^G_{\mathcal{F}^s,\mathcal{A}} > \delta^G_{\mathcal{F},\mathcal{A}}$, for another DAS $\mathcal{F}$*

The two conditions for a schedule to be a strongly (resp. weakly) SLP-aware DAS are:

Algorithm 1 An algorithm to verify if a TDMA slot assignment is SLP-aware.

```
    ▷ G: network topology, F: schedule, A: attacker
    ▷ δ: safety period, S: source
    ▷ Function returns boolean, a (violating) sequence, and a (capture) period
 1: function VerifySchedule(G, F, A, δ, S)
 2:     period, num := 0, 0
 3:     history := [0, ..., 0]_{1×A.H}
 4:     P := GenerateAllAttackerTraces(G, F, A, S)
 5:     while P ≠ ∅ do
 6:         i, pc := 0, Choose(P)
 7:         while i < |pc| do                                   ▷ pc = n_0 · n_1 ...
 8:             B := 1HopNsWithRLowestSlots(n_i, F, A.R)
 9:             if n_{i+1} ∉ {m|(m, n_i) ∈ E}      ▷ Going to an unheard location or not according to A.D
                    ∧ (S(n_{i+1}) ∉ B ∨ n_{i+1} ∉ A.D(B, history))  then
10:                 break
11:             if S(n_i) > S(n_{i+1}) then
12:                 period, moves := period + 1, 1
13:             else if moves = A.M then
14:                 break
15:             else
16:                 moves := moves + 1
17:             if n_{i+1} = S ∧ period ≤ δ then
18:                 return (False, pc, period)                  ▷ Captured within safety period
19:             if A.H > 0 then                  ▷ Only update history if the attacker is capable
20:                 history[num], num := n_i, (num + 1) mod A.H
21:             i := i + 1
22:         P := P \ {pc}
23:     return (True, ⊥, δ)
```

1. The schedule has to be a strong (resp. weak) DAS.

2. The capture time of $\mathcal{F}^s$ is greater than that of $\mathcal{F}$.

Definition 7 (δ-SLP-aware for $\mathcal{S}$)**.** *Given a network $G = (V, E)$, a (R, H, M, s_0, D)-$\mathcal{A}$ attacker, a safety period δ, a source $\mathcal{S}$ and a DAS schedule $\mathcal{F}^s$: $\mathcal{F}^s$ is* δ-SLP-aware[2] *for $\mathcal{S}$ in G in the presence of a (R, H, M, s_0, D)-$\mathcal{A}$ attacker if and only if* VerifySchedule$(G, \mathcal{F}, \mathcal{A}, \delta, \mathcal{S}) = (True, \bot, \delta)$*. Otherwise, if* VerifySchedule$(G, \mathcal{F}, \mathcal{A}, \delta, \mathcal{S})$ $= (False, pc, p)$*, then $\mathcal{A}$ captures $\mathcal{S}$ in G under $\mathcal{F}$ within p rounds using pc.*

Here, the decision procedure works in a way analogous to model checking where a counterexample is returned. Similarly, *pc* represents the counterexample in terms of the locations the attacker can visit before capturing the source. Further, if $\mathcal{F}^s$ is δ-SLP-aware for $\mathcal{S}$, then $\delta^G_{\mathcal{F}^s,\mathcal{A}} > \delta$.

3.3.4 Fault-Tolerance

DAS and convergecast in the presence of a number of node crashes are now provided

[2]SLP-aware will also be stated when δ is clear in the context.

Definition 8 (Fault-Tolerant DAS/convergecast)**.** *Given (i) a network $G = (V, L)$, (ii) a children function $\mathbb{C}$, (iii) a sink-distance function δ, (iv) a schedule Ξ, and (v) f node crash faults,* Ξ is a *fault-tolerant* strong DAS/weak DAS/convergecast in the presence of faults *iff there exists a schedule Ξ' such that*

- Ξ *is a strong DAS/weak DAS/convergecast in the absence of faults.*
- Ξ' *is a strong DAS/weak DAS/convergecast in the presence of faults.*
- $\Xi = \Xi'$

A fault-tolerant DAS captures the idea that, in spite of node crashes, the schedule retains its DAS/convergecast property. However, this may be difficult to achieve. Hence, it is necessary to provide a weaker notion of fault tolerance in Definition 9.

Definition 9 (Eventually Fault-Tolerant DAS/convergecast)**.** *Given (i) a network $G = (V, L)$, (ii) a children function $\mathbb{C}$, (iii) a sink-distance function δ, (iv) a schedule Ξ, and (v) f node crash faults:* Ξ has an *eventually fault-tolerant* strong DAS/weak DAS/convergecast Ξ' in the presence of faults *if and only if there exists a schedule Ξ' such that*

- Ξ *is a strong DAS/weak DAS/convergecast in the absence of faults.*
- Ξ *violates strong DAS/weak DAS/convergecast in the presence of faults.*
- $\Xi = \Xi'$ *in the absence of faults f*
- Ξ' *satisfies strong DAS/weak DAS/convergecast in the presence of faults.*

The idea of eventual fault tolerance in the presence of node crashes is that, temporarily, DAS or convergecast may be violated. The protocol will then eventually generate a new schedule that satisfies DAS. In the absence of faults, the original schedule is precluded from changing (i.e., adapting).

3.3.5 Fault-Tolerant SLP-Aware DAS

Before formalizing the notion of fault-tolerant SLP awareness, the meaning of a DAS/convergecast schedule being SLP-aware is presented.

Definition 10 (SLP Awareness)**.** *Given (i) a network $G = (V, L)$, (ii) a children function $\mathbb{C}$, (iii) a sink-distance function δ, (iv) a schedule Ξ, (v) a source node $\mathcal{S}$ and (vi) a safety period α, Ξ is an* SLP-aware *strong DAS/weak DAS/convergecast for $\mathcal{S}$ iff:*

1. Ξ *is a strong DAS/weak DAS/convergecast.*

2. *There is no capture path* c *in* G *for* $\mathcal{S}$ *under* Ξ*, where* $length(c) \leq \alpha$*.*

The notion of safety period is that it forces the attacker to capture the source within a number of (time) steps. Essentially, it rules out trivial solutions such as exhaustive search. Now, the meaning of fault-tolerant SLP awareness is provided.

Definition 11 (SLP Awareness and (Eventual) Fault Tolerance)**.** *Given (i) a network* $G = (V, L)$*, (ii) a children function* $\mathbb{C}$*, (iii) a sink-distance function* δ*, (iv) an SLP-aware schedule* Ξ*, (v) a source node* $\mathcal{S}$*, (vi) a safety period* α *and (vii)* f *node crash faults,* Ξ *is a* (Eventually) Fault-Tolerant SLP-aware *strong DAS/weak DAS/convergecast iff:*

1. Ξ *has an (eventually) fault-tolerant strong DAS/weak DAS/convergecast* Ξ' *in the presence of faults.*

2. *There is no capture path* c' *in* $G \setminus F$ *under* Ξ'*, where* $length(c') \leq \alpha$ *and* F *is the set of crashed nodes with* $|F| = f$*.*

It is shown in Chapter 6 that it is impossible to develop an SLP-aware schedule where any node can be the source. So the source must be known to develop an SLP-aware schedule.

Chapter 4

Experimental Setup

This chapter outlines the specifics for implementation and testing of algorithms in order to gather results.

4.1 OS and Simulator

A number of OSes exist for WSN and IoT use cases, many with different features and benefits.

4.1.1 TinyOS and TOSSIM

The first one examined was TinyOS (version 2.1.2) [85]. TinyOS is the oldest of the OSes examined, but that also means that it has been documented and supported for a long time. It has a number of design decisions that allow it to run on incredibly resource-constrained devices. It is implemented in a dialect of C known as nesC [59], which adds the ability to define components and interfaces and specify how they interact. This is done at compile time, which means component interactions are fixed at runtime.

TinyOS also has its own simulator known as TOSSIM [86]. TOSSIM only supports TinyOS to simulate, and simulates only one platform (MICAz). This is because TOSSIM is heavily integrated with TinyOS, replacing low-level components with some of its own during compilation which also means the CPU is not simulated on a per-instruction basis and an abstraction is created for how messages are sent over the wireless medium. This reduces the accuracy of the simulation, however allows the size of networks to increase up to 1000 nodes while still being performant.

Additionally, previous SLP work has been performed using TinyOS and TOSSIM which provides a framework containing a large amount of the features required (see Appendix A).

4.1.2 Contiki and COOJA

Contiki [45] is an alternative to TinyOS written in C. Contiki supports the concept of running multiple processes concurrently in the programming model [50] that can be resolved to suit the device at compile time.

Contiki can be simulated in COOJA [94]. COOJA also supports running other binaries, including those created using TinyOS. COOJA is a much more in-depth simulator than TOSSIM, as it also simulates the processor of the node which gives it the ability to run many different binaries made for different targets. Another benefit of this is that executing code advances a node's clock, unlike in TOSSIM where code execution takes no simulator time to perform. While COOJA strives for accuracy of simulation, this severely impacts the speed and scalability of simulations.

4.1.3 Choice

The chosen OS to use is TinyOS. This decision is mainly driven by being able to use a simulator that can scale, i.e. TOSSIM, which will only run TinyOS code. The algorithms created in this thesis will be simulated in a networks of varying sizes, including those considered to be large-scale deployments, which COOJA could not run at any acceptable speed.

Additionally, the existence of an operational SLP framework wrapping TinyOS finalised the decision.

4.2 Network Configuration

Throughout this thesis the networks to be tested are comprised of nodes in a square grid. Simulations are performed on network sizes of 7×7 (49 nodes), 11×11 (121 nodes), 15×15 (225 nodes), 21×21 (441 nodes) and 25×25 (625 nodes). This variety of sizes shows algorithm utility and levels of SLP provided across different network scales.

The sink node (the data collection point) is placed in the exact centre of these networks, while the source of messages is placed in one corner.

The separation between nodes is 4.5m vertically and horizontally. This distance allows for communication to occur in only vertical and horizontal directions (not diagonally) in the ideal communication model (communication models are explained in further detail in Section 4.3).

This network configuration has been selected largely due to the properties derived from its degree of connectivity:

- The connectedness of the network is sufficient to test the properties of the solutions while also remaining suitable for very resource-limited motes in terms of memory. The mote being simulated is the MICAz and the chosen OS is TinyOS (as described in Subsection 4.1.1). Due to TinyOS lacking support for dynamic memory allocation (which is understandable in such a resource-constrained environment), memory is statically allocated at compile time within the binary that is flashed to the device. The MICAz has 128K of program memory, which includes the statically allocated memory, severely limiting the capability of the device to store two-hop neighbourhood information (as required in Chapter 6 and Chapter 7) if the network is too connected. While this is a limitation of the chosen platform, it shows that these solutions can be employed on devices of extremely low cost compared with other motes within certain connectivity assumptions. For any use-case that involves a higher degree of connectivity, motes with increased memory can be used but would involve an increase in deployment cost. It is also important to note that memory usage is not increased based on network size, so scalability of these solutions is not compromised.

- An additional aspect of limiting the degree of connectivity is to provide a substantial network diameter for all simulated network sizes which provides an attacker model with more options for longer paths and provides an easier examination of their efficiency than if the network is highly connected, as the sink-source distance would be significantly reduced. For instance, the higher the degree of network connectivity, the fewer available paths exist to create a diversionary route for the attacker to follow.

- Debugging or analysing simulations containing between 49 and 625 nodes is a complex and time-consuming task. These processes can be simplified by providing a method of examination through visual aid. By providing a clean structure to the network and having clear limitations on potential recipients of message transmissions and limited options in attacker movement patterns, examination of the solution or attacker behaviour is far more intuitive to visualise and as such rationalise.

- Using a consistent network configuration as a control variable throughout this work allows for comparisons between more valuable aspects of the research, such as the varying solutions proposed in Chapter 5, Chapter 6 and Chapter 7.

- Finally, while network configuration is certainly useful to analyse, it is not the priority of this research. It has however been declared as a potential option of

Table 4.1: Low-Asymmetry LinkLayerModel Parameters

Name	**Value**
PATH_LOSS_EXPONENT	4.7
SHADOWING_STD_DEV	3.2
D0	1.0
PL_D0	55.4
NOISE_FLOOR	−105
S	[0.9 −0.7; −0.7 1.2]
WHITE_GAUSSIAN_NOISE	4

furthering this research with future endeavours in Section 8.2.

Additionally, examination of algorithms and other aspects of WSNs in the context of a grid topology has been performed in existing research [17, 60, 108, 109].

4.3 Simulation Parameters

The WSN simulation environment that is used in this work is TOSSIM, TinyOS's (version 2.1.2) own discrete event network simulator. Two communication models are compared, the first being an *ideal* model which sets a fixed signal receive strength when in range, making it almost 100% reliable. The second, named *low-asymmetry*, is generated using the LinkLayerModel provided with TinyOS and the parameters shown in Table 4.1. Low-asymmetry gives the simulation a small probability of packet losses and links becoming unidirectional, more akin to a real-world application. The noise model used is the first 2500 lines of the casino-lab noise file provided with TinyOS.

A safety period (the length of time that the source can be prevented from capture) is also provided. The time taken to reach the source in a protectionless TDMA DAS environment is $periodLength \times (\Delta(Sink - Source) + 1)$, where $\Delta(Sink - Source)$ is the number of hops between the sink and source, as this is the time it would take for the attacker to travel one of the shortest paths to the source. Therefore, an upper bound of $periodLength \times (\Delta(Sink - Source) + 1) \times 1.5$ is used. The reasoning behind selecting the factor of 1.5 is that the solutions in this work are designed to take the attacker on a diversionary route around the network before the attacker finally arrives at the source. An assumption is used that the location of the asset, and as such the source, will have changed by the time this route has been followed and the attacker will arrive at the stale source location, as such failing to capture the asset. Specifically using the factor of 1.5, when coupled with the previously defined network configuration, provides ample time for the attacker

to reach the source if the diversionary route has not been followed or the route was not sufficient enough to slow the attacker but will have expired if an adequate diversionary route was created and followed by the attacker.

4.4 Metrics

In order to gauge the effectiveness of algorithms, a number of metrics must be selected in order to compare them.

Capture ratio is largely considered to be the most important metric, as this is a strong indicator of the level of SLP that an algorithm provides. Capture ratio is the percentage of repeats where the attacker captured the source in a time less than the safety period.

Other metrics are used to examine the suitability of the algorithm for general use and any trade-offs that may have occurred while attempting to provide SLP. These metrics are used as necessary when the scenario requires them. They are: normal message send rate, normal message receive ratio, normal message latency and SLP message overhead.

4.5 Testbed Deployment

WSN testbeds are real-world deployments in a controlled environment. There are a number of testbeds open to academic use.

There are two main problems to selecting a testbed for this research. The first is compatibility with TinyOS, the OS used to develop implementations of the algorithms. Secondly, due to the nature of the algorithms, a good amount of spatial redundancy is a must between the source and the sink in order to create a diversionary route.

Due to a lack of many publicly-available testbeds, also combined with the above criteria, reduced the set of available testbeds to zero.

For example, Flocklab [87], a testbed with compatibility for TinyOS, looked promising. However, [25] shows that even broadcasting at the minimum power level, the network is still too small. Figure 4.1 shows the probabilities of message delivery along links in Flocklab at the lowest radio power (making the network as sparse as possible). The diameter of the network is not very large, leaving a short distance between source and sink as well as still being quite heavily connected. This is not suitable for the algorithms presented in the remainder of this thesis, as there is not much opportunity (not many paths available) to divert the attacker away from the source.

This leads me to conclude that without a larger, sparser testbed, it is not possible to test the algorithms in such an environment.

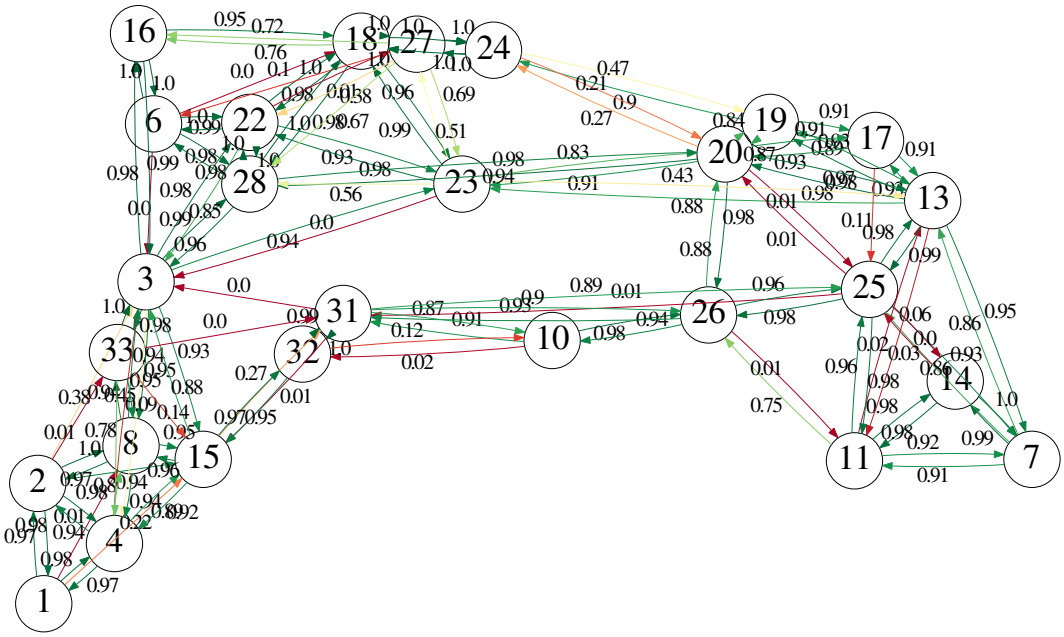


Figure 4.1: The probability of messages being delivered along links on the Flocklab testbed [25]

Chapter 5

Towards Optimal Source Location Privacy-Aware TDMA Schedules

Developing an optimal and valid TDMA schedule is known to be NP-complete [106]. However, the proposal is to add SLP as another optimization criterion in the design of optimal TDMA schedules. The aim is thus to generate SLP-aware slot assignments utilising an evolutionary method in order to produce schedules that are valid for the network and also provide SLP.

Genetic algorithms have been used for a wide variety of purposes in the sensor networks field. They have been used to find optimal parameters for routing protocols, such as LEACH [91], where it was used to determine weightings for combining multiple heuristics to determine which node became the next cluster head. The goal of this was to increase network lifetime by balancing energy loss between nodes more effectively. In [37], they went so far as to produce a new routing protocol created by a GA that is comparable to LEACH in order to use the least energy possible for those networks that harvest energy from the environment rather than batteries.

Genetic algorithms have also been used in the deployment of WSNs. The maximum coverage sensor deployment problem (MCSDP) is the problem of finding the minimum number of nodes required to cover a certain area [115]. Additionally, further work has been performed such that the deployment is augmented to find a solution that maximises the network lifetime by reducing energy requirements [92].

The allocation of TDMA time slots using genetic algorithm-related methods has previously been investigated [31, 44, 106] and finding an optimal TDMA schedule has been shown to be NP-complete [106]. So generating a TDMA schedule is a suitable problem for obtaining a near optimal result with GAs [6]. As the problem

exists in NP, a benefit is that there exists a polynomial-time method of checking a solution's validity which is necessary as an efficient method is required for use as a fitness function.

Genetic algorithms search the solution space to provide a good approximation of the optimal solution (should one exist). They face no restrictions on the variables to be optimised (for instance, requiring a continuous range of values) and are generally applicable to any optimisation problem. In fact, they can provide good solutions to a wide variety of optimisation problems assuming their characteristics are defined suitably. Due to this versatility, and as they have already been shown to produce positive results in generating TDMA schedules, adding a reasonable number of additional constraints to the problem to provide both DAS and SLP was determined to be a suitable adaptation of the existing methods. Once discovered that this method could indeed produce the desired schedules, these properties were further developed into the following five contributions:

1. The SLP problem is mapped onto a GA problem.
2. Suitable *crossover*, *mutation* and *selection* operators are presented to expedite the generation of optimised schedules.
3. The notion of Pareto optimality is used to compare the various generated schedules using two different *fitness functions* for analysis.
4. Simulations are performed in both ideal and realistic environments, showing metrics about the generated solutions such as near optimal capture ratio and high packet delivery ratio.
5. Those solutions that lie on the Pareto frontier are examined and the optimal solution of those generated for a specific network configuration is determined.

The chapter is structured as follows. The GA and its operators are detailed in Section 5.1. Section 5.2 explains Pareto efficiency and its utility in this context. Section 5.3 describes how the GA generates solutions, the experiments to be run and any further experimental information not already defined in Chapter 3. The results from the simulations are shown and discussed in Section 5.4 and a summary of this chapter's contributions is presented in Section 5.5.

5.1 Genetic Algorithm

5.1.1 Algorithm

The algorithm selected for this work is the generic and standard genetic algorithm using non-standard and optimised genetic operators for the problem domain. One reason for this selection is so a genetic algorithm can be built to the required specification and optimised for the SLP problem.

The selected algorithm is *not* a true, multi-objective genetic algorithm. [34] finds that using a scalar combination of fitness scores speeds up computation over using a multi-objective algorithm, at the disadvantage of being unable to discover concave Pareto fronts. Due to the comparison of the two different fitness qualities that will be evaluated to determine the Pareto front (further explained in Section 5.2), a concave Pareto front is not expected and as such using a scalar combination of fitness scores to decrease computation time was determined to be the correct approach. However, the usage of a more advanced, true multi-objective GA, NSGA-II [38], is examined as a comparison.

5.1.2 Genome Representation

For this problem, the representation must contain a graph $G = (V, E)$, where E is the set of edges between nodes in V who are within communication range. Each node in V is also labelled with a slot value, the number of hops from the sink, a parent node and a set of child nodes.

The reason that this representation differs from that used by [106] is that it is incapable of portraying the additional constraints required by DAS, so a full graph representation must be used instead.

5.1.3 Genetic Operators

As described in [106], providing operators with a context of the problem domain and forcing the creation of valid solutions after each operator's function can aid in providing better solutions using less generations at a performance cost at each step. The operators described below each produce genomes that are valid DAS solutions.

Initialisation

The initialisation function is required to create an individual to be included in the initial population.

Algorithm 2 performs the network initialisation. Within the algorithm, the first stage is to initialise all attributes of each node with sensible defaults. The

parameter *maxSlots* is used to provide a number of *pseudo slots* for the genetic algorithm to assign during operation. Providing a number too large will result in more slot values being different (thus more slots used in the final solution) where providing too few will result in the genetic algorithm struggling to find a solution. In the final solutions, all pseudo slot values are normalised to their minimum range. The next stage creates a DAS parent-child tree from the nodes, changing slot values to ensure the genome conforms to the DAS specification. The function, SetParent, sets a node's parent and also adds the node to the parent's *children* set.

Algorithm 2 Algorithm for initialising a genome

```
    ▷ g: genome, sink: sink node, maxSlots: number of pseudo slots
 1: function Initialise(g, sink, maxSlots)
    ▷ First initialise all nodes with some values
 2:   for all node ∈ g do
 3:     if node = sink then node.slot := maxSlots
 4:     else node.slot := RandomInteger(1, maxSlots − 1)
 5:     node.hop := ShortestPathLength(g, node, sink)
 6:     node.parent := ⊥
 7:     node.children := ∅
 8:   repeat := 1
 9:   while repeat do
10:     repeat := 0
        ▷ Assign a parent to each node without one
11:     for all node ∈ {n | n ∈ g \ {sink} ∧ n.parent = ⊥} do
12:       slotChanged := 0
13:       parentChoice := GetPotentialParents(g, node)
14:       SetParent(g, node,Choose(parentChoice))
15:       if node.slot ≥ node.parent.slot then
16:         node.slot := node.parent.slot − 1
17:         slotChanged := 1
          ▷ Resolve collisions
18:       nhs := GetTwoHopNeighbourhoodSlots(g, node)
19:       while node.slot ∈ nhs do
20:         node.slot := node.slot − 1
21:         slotChanged := 1
22:       if slotChanged then
23:         for all c ∈ node.children do
24:           SetParent(g, node, ⊥)
25:           repeat := 1
26:   return g
```

Crossover

The crossover operator is used to combine two parent individuals together to create two children which incorporate features from both parents.

The crossover operation is contained within three functions in Algorithm 3: Crossover, OneCrossover and CheckCrossover. Crossover is a wrapper function that simply selects the node which will be the head of the crossover and

returns the product of performing crossover for each child. OneCrossover gets the subnetwork that is being introduced to the child and overwrites those elements in the new child genome. CheckCrossover is where any issues pertaining to the validity of DAS get resolved, by starting at the crossover node and recursing through all descendents.

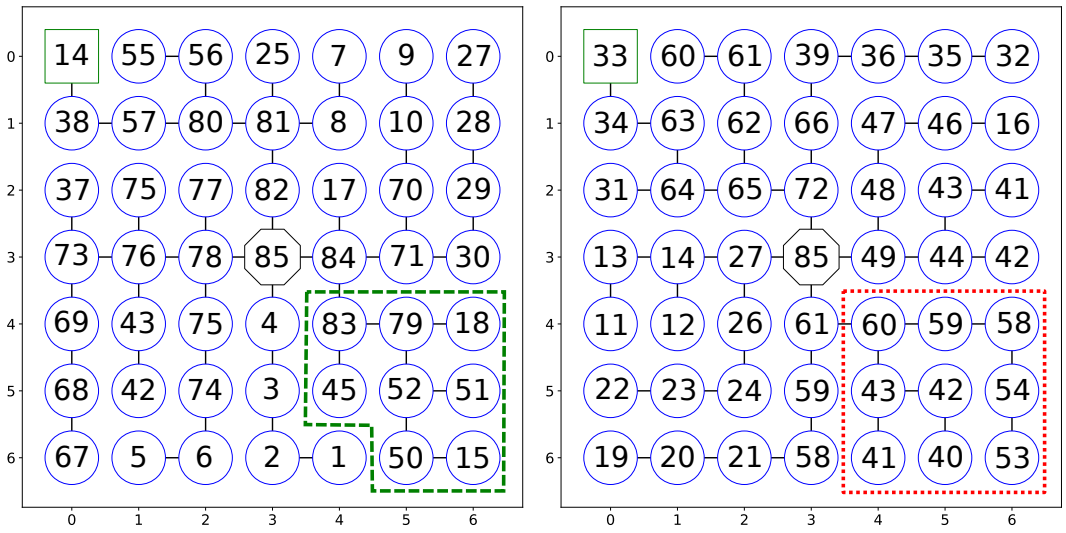
Figure 5.1 shows an example execution of this method. Starting with a father and mother genome, the algorithm begins by selecting the crossover node, which in this case is at the coordinates $(4, 4)$. Then, the DAS parent-child tree structure from the father is transplanted into a clone of the mother solution (the daughter) and the same occurs for the mother to the father (creating the son). The slot values and the DAS structure can then be altered to ensure DAS validity remains in the child genomes.

Algorithm 3 Crossover two genomes

```
    ▷ father: father genome, mother: mother genome, sink: sink node
 1: function Crossover(father, mother, sink)
 2:     son, daughter := father, mother
 3:     crossoverNode := Choose(father \ {sink})
 4:     son := OneCrossover(son, mother, crossoverNode)
 5:     daughter := OneCrossover(daughter, father, crossoverNode)
 6:     return son, daughter

 7: function OneCrossover(g1, g2, crossoverNode)
 8:     subnetwork := GetDescendents(g2, crossoverNode)
 9:     for all node ∈ subnetwork do
10:         g1[node].slot := g2[node].slot
11:         if g2[node].parent ∈ subnetwork then
12:             SetParent(g1, node, g2[node].parent)
13:         else                                  ▷ This occurs with the crossoverNode
14:             SetParent(g1, node, ⊥)
15:     CheckCrossover(g1, crossoverNode)
16:     return g1

17: function CheckCrossover(g, node)
18:     if g[node].parent = ⊥ then
19:         parentChoice := GetPotentialParents(genome, node)
20:         SetParent(g, node,Choose(parentChoice))
21:     if g[node].slot > g[node].parent.slot then
22:         g[node].slot := g[node].parent.slot
23:     nhs := GetTwoHopNeighbourhoodSlots(g, node)
24:     while g[node].slot ∈ nhs do
25:         g[node].slot := g[node].slot − 1
26:     for all child ∈ g[node].children do
27:         CheckCrossover(g, child)
```



(a) Father genome (b) Mother genome

(c) Son genome (d) Daughter genome

Figure 5.1: An example of performing crossover at node $(4, 4)$

Mutation

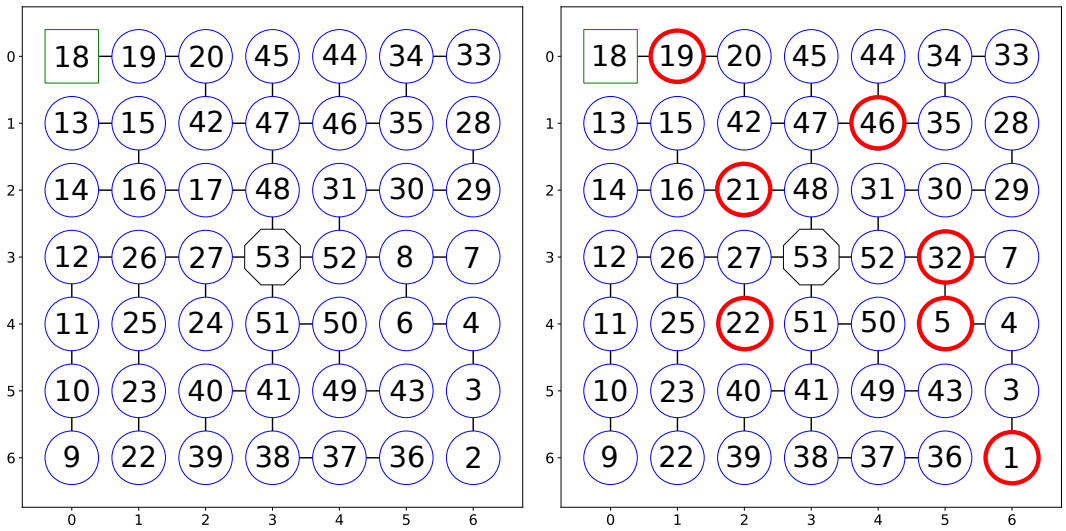
The mutation operator is an attempt to further introduce diversity into the population to prevent stagnation.

Algorithm 4 begins by defining a range in which the new slot value should be chosen. The $maxDiff$ parameter denotes the maximum range in which a slot value can be altered. Should the node have no children, the slot value is reduced by one, which repeats should there be a slot collision in the two-hop neighbourhood. The reason for this is to prevent a large variety of different slot values at the edges of the network. Otherwise, the low end of the slot range is refined to be within the acceptable range that the children of the node set (i.e. a parent node cannot have a lower slot value than its children). A new slot is then attempted to be randomly selected (avoiding slot collisions) subject to the parameter *attempts* which limits the number of times this is tried before exiting with failure.

Figure 5.2 shows an example of the algorithm in operation. For the example, the mutation rate is set to 10%, which is too high for normal operation but useful to see the effect of the operator. All cases of the algorithm can be seen, such as nodes with no children being reduced minimally (at coordinates $\langle 6, 6\rangle$ and $\langle 2, 4\rangle$ in the figure), nodes with children that have a range of slots to select from (at coordinates $\langle 2, 2\rangle$, $\langle 5, 3\rangle$ and $\langle 5, 4\rangle$) and finally nodes that have children but failed to change slot (at coordinates $\langle 1, 0\rangle$ and $\langle 4, 1\rangle$).

Algorithm 4 Randomly alter slot values within DAS constraints

```
    ▷ g: genome, node: node to mutate
    ▷ maxDiff: the maximum range of values to select new slot from
    ▷ attempts: the number of attempts to make at mutating the slot
 1: function MUTATE(g, node, maxDiff, attempts)
 2:   highest, lowest := node.parent.slot, highest − maxDiff
 3:   nhs := GETTWOHOPNEIGHBOURHOODSLOTS(g, node)
 4:   if node.children = ∅ then
 5:     node.slot := highest − 1
 6:     while node.slot ∈ nhs do
 7:       node.slot := node.slot − 1
 8:   else
 9:     for all child ∈ node.children do
10:       if child.slot > lowest then
11:         lowest := child.slot
12:     while attempts > 0 do                ▷ If attempts reaches zero, mutation failed
13:       newSlot := RANDOMINTEGER(lowest + 1, highest − 1)
14:       if newSlot ∉ neighbourhoodSlots then
15:         node.slot := newSlot
16:         break
17:       else
18:         attempts := attempts − 1
```



(a) Before mutation (b) After mutation

Figure 5.2: An example mutation using algorithm 4

Selection

The selection operator is used to select solutions from the new population to be entered into the next generation.

Standard tournament selection will be used where the number of solutions in each tournament is 2. Algorithm 5 shows the tournament selection method for a variable value of the number of solutions per tournament, t.

Algorithm 5 Select an individual from the population using tournament selection

▷ *pop*: population, t: number of genomes to compete in each tournament
1: **function** SELECT(pop, t)
2: $best := pop[\text{RANDOMINTEGER}(0, pop.size)]$
3: **if** $t = 1$ **then return** *best*
4: **for** $i := 2 \rightarrow t$ **do**
5: $next := pop[\text{RANDOMINTEGER}(0, pop.size)]$
6: **if** FITNESS($next$) > FITNESS($best$) **then** $best := next$
7: **return** *best*

5.1.4 Fitness Functions

The fitness function is used to assign a numerical *worth* to each individual in the population based upon its perceived quality. This score is then used by the selection operator in order to choose the best individuals for crossover and mutation.

The results from two different fitness functions will be examined. Algorithm 6 shows the first fitness function, which simply provides a value in the range $[0, 1]$ depending on the total number of slots used to generate the solution, where 0 would

be given if each node had its own time slot (the worst case) and 1 if each node had the same slot, which is not possible due to the DAS constraints. Therefore, the best achievable score for this function is unknown. Additionally, the function returns 0 if the attacker path intersects the source, as the solution does not provide SLP.

The second function (Algorithm 7) performs the same as the first with the additional optimisation of making the attacker move as far away from the source as possible. With the first function, the attacker path can be liable to take the attacker closer to the source. This is undesirable, as in the case of link failures and node crashes along the path that it is desired that the attacker takes, the attacker could leave the predicted path and travel to the source, thus capturing the asset. In this chapter, only the occurrence of link failures due to noise on the wireless channel will be examined. This function also provides a value in the range $[0, 1]$, with 1 being scored if the path leads the attacker as far away as allowed by the network. By using the parameter w, a weighting is provided to emphasise the further distance aspect over the regular total slot reduction. In this work, $w = 0.25$, meaning that slot usage is weighted at 75% and the distance is weighted 25% of the total fitness score returned.

The reason for comparing two fitness functions is that the second provides a better solution in an unreliable network, but it is unknown how this will affect the slot usage reduction process.

Algorithm 6 Slot usage fitness

```
    ▷ g: genome, src: source node
1:  function SlotUsageFitness(g, src)
2:      if src ∈ AttackerPath(g) then return 0
3:      uniqueSlots := {node.slot | node ∈ g}
4:      return (|g| − |uniqueSlots|) / |g|
```

Algorithm 7 Distance from source and slot usage fitness

```
    ▷ g: genome, source: source node
    ▷ w: the weighting for attacker paths end points distance to the source
1:  function DistAndSlotUsageFitness(g, src, w)
2:      if src ∈ AttackerPath(g) then return 0
3:      endNode := AttackerPathEnd(g)
4:      dist := ShortestPathLength(g, src, endNode)
5:      diameter := NetworkDiameter(g)
6:      distScore := w × (dist / diameter)
7:      slotScore := (1 − w) × SlotUsageFitness(g, src)
8:      return distScore + slotScore
```

5.2 Pareto Efficiency

After running the genetic algorithm, multiple competing solutions will be generated. Since the concern is maximising the level of source location privacy while minimising the number of slots and maximising the path distance between a source and the end point of an attacker, this search process is a multi-objective optimisation process. As such, no universal definition of an optimum can be given. One solution for an appropriate definition of optimality in multi-objective optimization was given by Vilfredo Pareto in 1896. This definition expresses that a solution is *Pareto-optimal* if there exists no other feasible solution which would decrease some objective without causing a simultaneous increase in at least one other objective [84].

More formally, consider a set of n solutions $s_i \dots s_n$, with a solution s_i being a vector of m attributes, i.e., $s_i = \langle a_i^1 \dots a_i^m \rangle, a_i^j \in \mathcal{R}$. A solution s_i is Pareto-optimal if there is no other feasible solution s_j where $j \neq i$ such that, for a given set of m utility functions $U_1 \dots U_m$, $\langle U_1(a_j^1), \dots, U_m(a_j^m) \rangle \geq \langle U_1(a_i^1), \dots, U_m(a_i^m) \rangle$ with at least one attribute a_j^k such that $U_k(a_j^k) > U_k(a_i^k)$. On the other hand, if such a solution s_j exists, then s_j is a Pareto improvement on s_i.

Given a set of feasible solutions (and a way of valuing them), the *Pareto frontier* is the set of choices that are Pareto efficient. Formally, given a solution space $\mathcal{S}$, the Pareto frontier of S, denoted by $P(\mathcal{S})$ is $P(\mathcal{S}) = \{s \in \mathcal{S} | \forall s' \in \mathcal{S}$ s.t s' is not a Pareto improvement on $s\}$.

Using Pareto dominance allows a comparison of many solutions, where one can easily analyse the tradeoffs between selecting any one solution compared with another.

5.3 Experimental Setup

In this section the environment and methods used to generate the results in Section 5.4 are described. This genetic algorithm[1] was implemented using Python, the Pyevolve genetic algorithm framework [96], and the graphing library NetworkX [65]. An implementation of the multi-objective genetic algorithm NSGA-II [38] was utilised.

5.3.1 Genetic Algorithm Parameters

The main parameters for a genetic algorithm are *elitism*, *crossover rate*, *mutation rate*, *population size* and *total generations*.

[1]The source code for the genetic algorithm can be found at `github.com/jack-kirton/slp-tdma-das-genetic`

Elitism is the number of copies of the best solution from the previous generation that should be entered into the next generation unaltered (i.e. no crossover or mutation). A value of 1 was chosen for this parameter as taking a single copy of the best solution from the previous generation to the next allows for the genetic algorithm terminating, ensuring that at least one solution that is supposedly good is available to present at the end of the algorithm.

The values of crossover rate, mutation rate and population size are set to the Pyevolve defaults of 90%, 2% and 80 respectively. The values chosen for parameters are sensible defaults. Finding the most optimal parameter configuration is left as an exercise for future work. Total generations for this work is capped at 200, which was shown through testing to be a reasonable compromise between time taken and quality of result. For results that optimise each fitness score further, more generations can be used at the cost of time.

5.3.2 Network Configuration

This is largely as described in Section 4.2 with the exception that only the 11×11 (121 nodes) network size is used. Utilising only this size of network to evaluate was done as increasing the network size increases both the genetic algorithm schedule generation time, the simulation time and even the number of simulations that need to be run should multiple different sizes be evaluated. Limiting the network to this size enabled gathering of a substantial number of solutions from the GA and to simulate them all with adequate repetitions to improve accuracy of the metrics. An even smaller network could have been used but the selected size balances computation time while still being large enough to produce interesting results.

5.3.3 Simulation Parameters

These are largely as described in Section 4.3, with only the number of repeat simulations differing. A total of 100 solutions per fitness function (for a total of 200) will be created by the genetic algorithm. Each of these will have a unique slot assignment compared with the others, providing a different operating scenario between each solution. To gather results about the efficacy and efficiency of the solutions, approximately 600 runs of each solution will be performed in the simulator each for two different communication models, providing a total of $240,000$ simulation runs.

5.3.4 Algorithm Parameters

Most parameters for the algorithm are provided by the generated genetic algorithm solution (i.e. number of slots). Two that are not provided are the dissemination and slot period lengths. These values were set to 0.5 seconds and 0.1 seconds respectively. During the simulation, no data will be transmitted in the dissemination period as all nodes are aware of the data they require at compile time. Time synchronisation would normally be used in this period but this is not modelled by the simulator. Given the number of slots and the dissemination/slot periods, the rate at which the source sends messages (the source period) can be calculated as the desire is for the source to send a single message per TDMA period.

5.4 Results

5.4.1 Expected Outcomes

Four metrics from the TOSSIM simulations will be examined: messages sent, capture ratio, message receive ratio and latency. A comparison will be made between combinations of fitness function and communication model for each solution produced by the GA.

The number of messages sent is expected to be similar across all solutions. There will likely exist variation between solutions depending on the total number of slots utilised, as this implicitly defines the length of a TDMA period. Better slot utilisation will result in a quicker TDMA period, resulting in a higher message send rate.

Capture ratio should be 0% for both fitness functions with the ideal communication model. This is because there should be no radio interference causing the attacker to leave the predicted path. However, with the low-asymmetry communication model, because of the noise and varying link qualities, both fitness functions should show varying capture ratios. Between the two fitness functions, it is expected that the slot usage fitness function will have more solutions of a higher capture ratio than that of slot usage and path distance.

Message received ratio is the percentage of normal messages sent by the source that were received by the sink. The message received ratio should be close to 100% for the ideal communication model due to perfect link quality, whereas low-asymmetry will provide slightly less.

Latency is the amount of time taken for normal messages sent by the source to reach the sink. The expectation is that the messages will travel from source to sink in a maximum time of the TDMA period (excluding the dissemination period).

This is due to the design of DAS, where messages should be able to travel from any point in the network to the sink in a single period due to the path of increasing slot assignments.

Those solutions selected from the Pareto frontier will be examined in further detail, finally determining the optimal solution for the configuration.

5.4.2 Genetic Algorithm

Figure 5.3 shows two example solutions produced by the GA, each using a different fitness function. Figure 5.3a shows a solution using the standard slot usage fitness function and Figure 5.3b shows a solution using slot usage and optimising the attacker path end's distance from the source.

In both examples contained in Figure 5.3, blue circles are normal nodes with the source node being labelled as a green square in the top left and the sink node being placed in the centre as a black octagon. The yellow hexagons are the nodes that the GA has predicted the attacker will follow through the network, known as path nodes. The reason that the attacker will follow this path is that each next node in the path has the lowest slot assignment in the surrounding one-hop neighbourhood, which means the attacker will hear from that node before any others. The connecting lines between the nodes show the DAS parent-child hierarchy, which is a tree structure rooted at the sink node. As previously stated, to ensure DAS-compliance, each node (bar the sink node) must have a parent that has a higher slot than itself and the connecting lines show that DAS has been preserved.

What isn't shown clearly by the examples is that the GA produces attacker paths that cause the attacker to oscillate between the end two nodes of the path indefinitely. This is because each of the two nodes at the end of the path has the lowest slot in the one-hop neighbourhood surrounding the attacker.

Examining the attacker paths visually, the difference between the two fitness functions can clearly be seen. As expected, including an optimisation to have the attacker travel as far away from the source as possible has caused the genetic algorithm to produce solutions that do just that, forcing the attacker into the opposite corner of the network (or thereabouts).

Figure 5.4a shows the cost of the additional fitness constraint, where *slot* is the original fitness function and *dist* is the function with the distance constraint. The plot shows generations against the number of slots used by the best candidate of that generation. The plot was produced using averages from 100 runs of each fitness function. This plot shows that there is an increase in the number of slots required to support the additional distance optimisation. However, the difference in total slots used is still a small value, although this would rise on more connected networks with

larger numbers of nodes.

5.4.3 Pareto Efficiency

Figure 5.4b shows a plot of solutions comparing the slot usage and path distance fitness functions. Each point on the plot shows solutions at various generations. The Pareto frontier can be seen as the maximal boundary over all of the solutions. The plot shows that solutions can easily maximise the path distance fitness function, which is intentional as the path distance can only be as large as the network allows, thus gaining the maximum score. The slot usage fitness function does not allow for a maximum value to be obtained, causing an apparent maximum of approximately 0.8, on which the Pareto frontier lies. Along the top of the plot, the solutions have maximised the path distance score but slot fitness increases as the solution is further to the right. This means every solution on the path-dist = 1.0 line that has a solution to its right is completely dominated by this solution. This means that along this line, the only solution that should be selected is the farthest to the right. With this solution and the others on the plotted line make up the pareto frontier. The two most promising solutions are `slot43` and `dist67`.

5.4.4 Simulations

The graphs in Figure 5.5 plots metrics from the output of the TOSSIM simulations, with each column being an output from the GA.

Firstly, Figure 5.5a shows a plot of messages sent per node per second. Using the rate of message sending rather than total messages sent is a better metric as simulations may not last the full safety period if the attacker happens to capture the source. The rate messages are sent appears similar across fitness functions and identical when changing communication model. This indicates that the differences are caused by slot utilisation, as predicted.

Figure 5.5b plots the capture ratio of solutions. The plot shows that using the ideal communication model, capture ratio is very low. With low-asymmetry, the capture ratio dramatically increases for both fitness functions, however the additional path distance constraint provides an improvement in both the number of high capture ratio solutions and the level of capture ratio on these solutions. Low-asymmetry causes these issues as the attacker leaves the path that the GA predicts it will follow, due to failing links as well as additional links that the GA did not expect (diagonal communication). This follows the predicted outcome.

Figure 5.5c plots the message receive ratio of messages sent from the source that are received by the sink. The ideal communication model performs with almost

100% receive ratio while low-asymmetry has a slightly lower rate due to the probability of link failures.

Finally, Figure 5.5d plots the latency of normal messages sent by the source to be received at the sink. The predicted outcome for all solutions is that the latency is bounded by a single TDMA period, excluding dissemination period. However, a handful of solutions of the low-asymmetry communication model do not abide by this prediction, appearing to approximately double the length of time taken for the message to travel from the source to the sink. A potential reason for this is that the normal messages are generated by the application layer, starting the latency timing, at a point after the transmission slot has ended before the start of the next period. This could lead to a near doubled latency for the message.

The two solutions making up the Pareto frontier are compared in Figure 5.6, using the low-asymmetry communication model. Messages sent and normal message latency differ between the two solutions due to the total number of slots used in the network. `dist67` is better in this regard, using less slots results in messages traversing from the source to the sink faster than the `slot43` solution. The receive ratio is very consistent between the two solutions meaning either is a good choice. Finally, capture ratio, while still very small values for both solutions, shows a slight improvement of `dist67` over `slot43`. With these metrics, it appears that the optimal solution for this configuration of all generated solutions is `dist67` as solutions not creating the Pareto frontier where eliminated followed by a metric analysis of those that remained.

5.4.5 Comparison with NSGA-II

In addition to utilising the Pyevolve framework, early tests were performed using an implementation of NSGA-II [38]. NSGA-II is a multi-objective genetic algorithm that is capable of producing a Pareto frontier by analysing solution domination over one another. The group of solutions that dominate all others but not each other is the Pareto frontier.

Unfortunately, as Figure 5.7 shows, the Pareto frontier is comprised of a large number of copies of exactly the same solution. It is unclear as to exactly what causes this lack of diversity in the population, however the selection method is the likely culprit, as the other genetic operators function well with the other GA. NSGA-II uses random tournament selection and compares rank followed by crowding distance to determine the better solution. The use of a crowding distance is supposed to aid in the selection of more diverse solutions but this does not appear to be the case for this specific usage.

Due to this selection issue, it also causes early stagnation of the population.

Table 5.1: The source and safety period combinations for DynamicSPR and ILPRouting in seconds

Source Period	Safety Period
0.25	4.57
0.5	8.64
1.0	16.85
2.0	33.76

Inevitably, this causes NSGA-II to underperform compared with the implementation of the other GA.

5.4.6 Comparison with other SLP solutions

This is the first GA-based TDMA scheduling for SLP in WSNs, making comparisons against other such schedules difficult. However, for the sake of completeness, comparisons are made using the best solution from the GA, with two other state-of-the-art algorithms that provide SLP at the routing level, namely *DynamicSPR* [24] and *ILPRouting* [21]. Those two protocols are instances of two classes of SLP protocols: (i) spatially-aware protocols and (ii) temporally-aware protocols [22] respectively. Due to the nature of these different protocols, different metrics will be applicable for comparison. Specifically, for spatially-aware protocols, the metric that is most important will be *message* overhead, whereas *latency* is the most important for temporally-aware protocols.

The same network size (11×11) and configuration (source in the top left and sink in the centre) was used for both algorithms to compare with the results gathered in this chapter. Further, the same communication (the ideal model) and noise models (casino-lab provided with TOSSIM) where used for all experiments. The source period and safety period parameters used for these algorithms are shown in Table 5.1. The source and safety periods as well as additional parameters for each algorithm are the same as those used in their respective papers, as these were proposed as the best parameters for providing the lowest capture ratio.

Since DynamicSPR and ILPRouting target the routing level and the GA-based solution targets the MAC level, the attacker models used during the comparison are addressed: the attacker model for the GA solutions has knowledge of the length of the TDMA period and moves upon hearing the first message during a given period, whereas the other routing protocols use sequence numbers to ensure that attackers are moving in a way as to not follow stale data, again following the first message heard. These different implementations result in very similar behaviour from the

attacker models.

Capture ratio: The main metric that will be used for comparison is the capture ratio. Figure 5.8 shows the capture ratios for the different source periods. The plots show that DynamicSPR has a capture ratio consistently below 0.5% while that of ILPRouting is higher at approximately 1%. The plots show the most optimal solution chosen from the GA-generated solutions, `dist67`, as a comparison. These comparisons are made under the ideal communication model, providing all three algorithms with a near-perfect environment. As the plots show, `dist67` has a 0% capture ratio which is better than both DynamicSPR and ILPRouting. Considering `dist67` capture ratio is based upon indefinite prevention of the attacker reaching the source (at least until the simulation time upper bound) while the other protocols are based on a safety period, this is an additional benefit of the GA-generated solutions.

Number of messages sent: Figure 5.9a observes that the number of messages sent by the GA solution is significantly lower than DynamicSPR. Since DynamicSPR requires fake messages to be sent to be efficient, the number of messages will increase over the GA solution which only sends application messages.

Latency: Figure 5.9b shows that the latency of the GA solution is approximately equivalent to ILPRouting with a source period of one. ILPRouting requires messages to be buffered and aggregated together before being sent, explaining the cause of the latency. This result is positive because the nature of TDMA usually requires higher latency due to dividing the medium into set time slots. This shows that the GA is optimising for minimal slots correctly and is comparable with ILPRouting.

5.4.7 Overview of Results

Firstly, some output from the GA was presented that showed the differences between the two fitness functions. This clearly displayed the effectiveness of the optimisation process as the distance fitness function created a path that lead the attacker into the opposite area of the network. Simulations were performed on 100 solutions for each fitness function using ideal and more realistic (low-asymmetry) communication models. This showed that for an ideal scenario, the solutions performed with an almost 0% capture ratio for every solution. Using the more realistic model, the effect of additional links and link failures in the network can be seen as both fitness functions have increased capture ratio. However, the distance fitness function outperformed the other on average.

The two solutions that lie on the Pareto frontier were then examined in further detail. This allowed us to select the best solution from the 200 solutions (named `dist67`).

Comparisons were made between the GA generation process and the high-

performance true multi-objective GA known as NSGA-II but found that it caused early stagnation in the population of solutions, providing inferior results.

Finally, the GA method was compared with two state-of-the-art algorithms, DynamicSPR [24] and ILPRouting [21]. Comparable capture ratios under the ideal communication model are shown. However, the best solution selected outperformed both algorithms in terms of capture ratio, while providing positive results in the other metrics shown.

5.5 Conclusion

In this chapter, the objective was to generate near-optimal SLP-aware TDMA schedules using evolutionary methods. A number of contributions have been made:

1. The SLP problem has been mapped to a GA problem that solves for a certain class of attacker (i.e. the distributed eavesdropper) producing suitable crossover, mutation and selection operators in the process.

2. Alongside producing SLP-aware schedules, two optimisation criteria were introduced; that of schedule latency and the attacker path leading away from the source. Thus two fitness functions were provided, one that only reduced schedule latency and one that combined schedule latency with having the endpoint of the attacker path as far from the source as possible.

3. The concept of Pareto optimality was used to compare resulting SLP-aware schedules with the two optimisation criteria of schedule latency and attacker path.

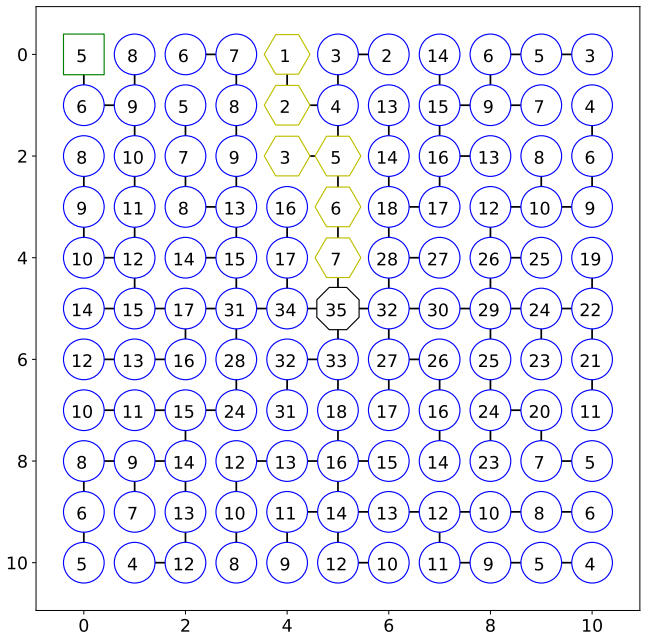
4. Simulations were performed with both reliable and unreliable communication models for all generated schedules to show their viability, providing a very low capture ratio and high delivery ratio to the sink.

The advantages of this proposed method are the near-optimal capture ratio coupled with the attacker path creation that occurs at no additional message overhead. Additionally, slot usage is optimised to reduce the total amount used, decreasing network latency.

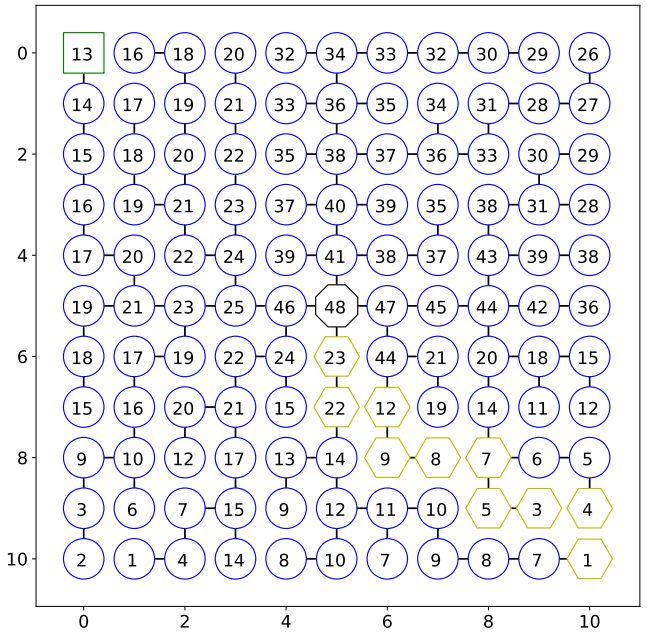
A downside of this approach is that the GAs typically run offline making it unsuitable for WSNs that determine a route online to adapt to the dynamism of real networks, such as when confronted by asymmetric links or link failures. The static slot assignment produced by the GA does not compensate for these issues which cause the attacker to leave the predicted path and travel along an unexpected route, usually increasing the capture ratio, While some online GA implementations

exist, due to a sensor node's limited resources this is not usually feasible. Having a more powerful sink node calculate the schedule is also possible, with techniques such as network reprogramming used to deploy it but these also tend to be expensive in terms of energy.

This conclusion presents the need to generate the schedules online in order to adapt to changing network conditions.

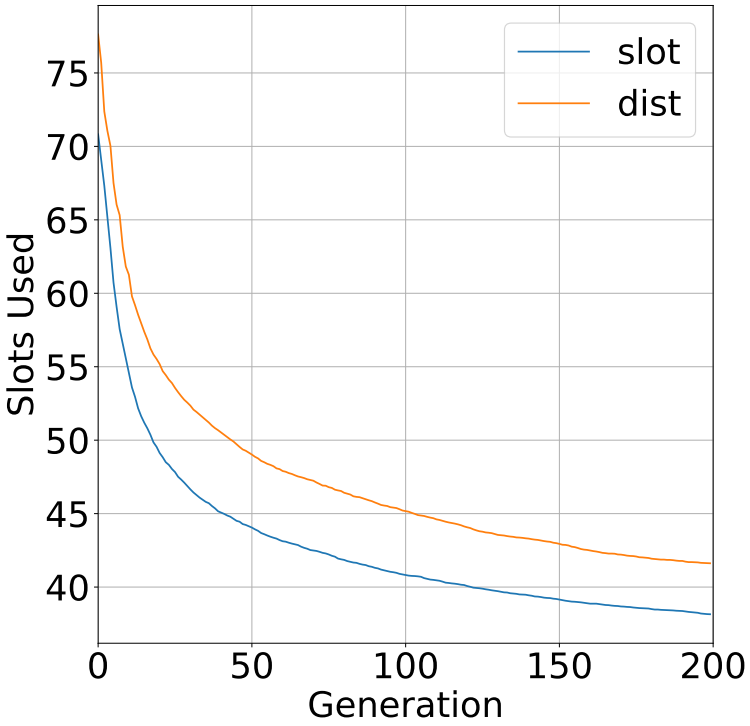


(a) Slot usage

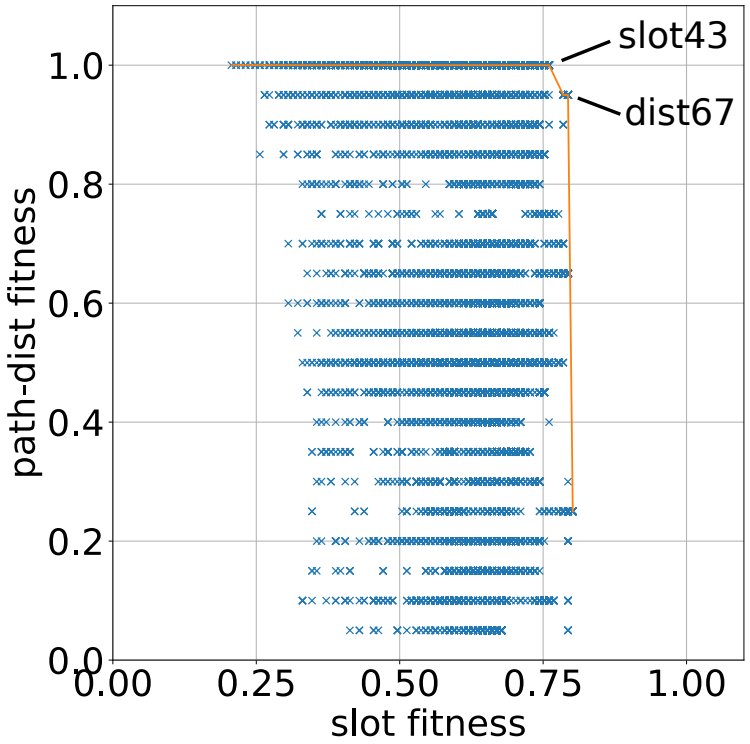


(b) Slot usage and distance from source

Figure 5.3: Example solutions produced by the GA using different fitness functions

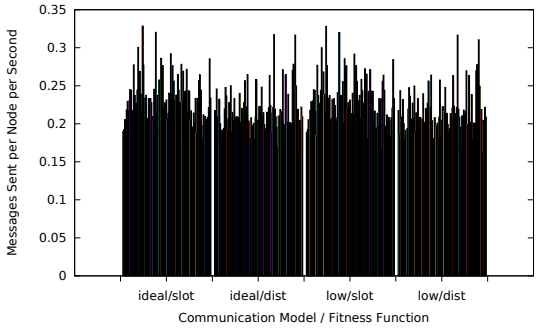


(a) A comparison of the slots used for the two fitness functions

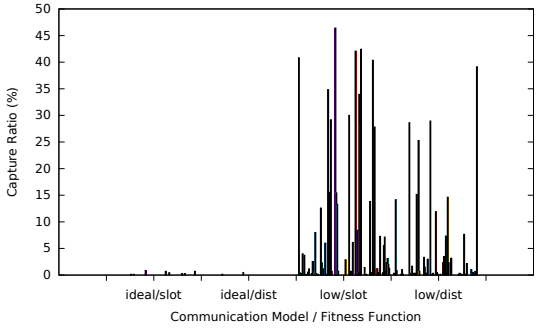


(b) Slot usage fitness versus path distance fitness

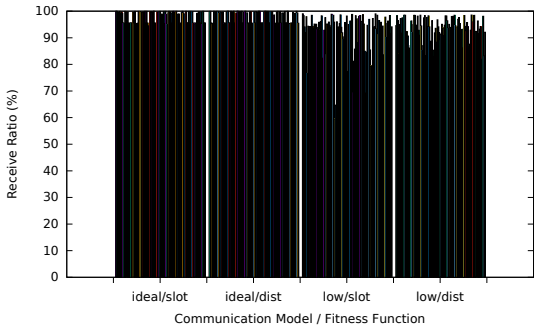
Figure 5.4: Comparison of fitness functions



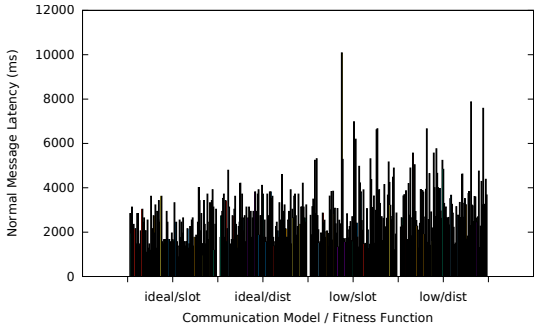
(a) Messages Sent per Node per Second



(b) Captured

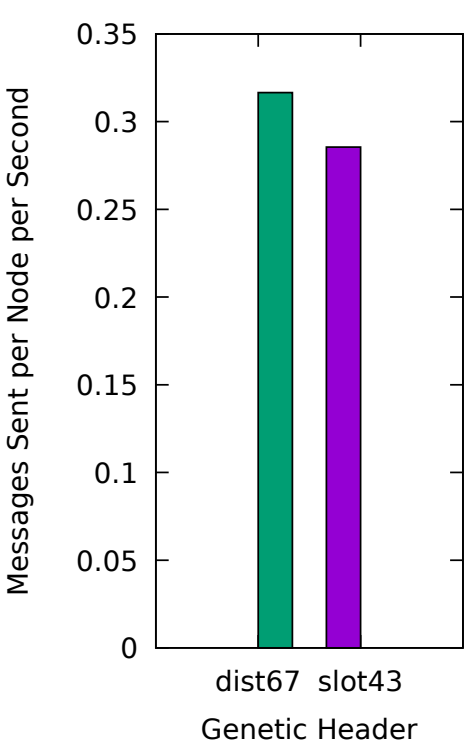


(c) Receive Ratio

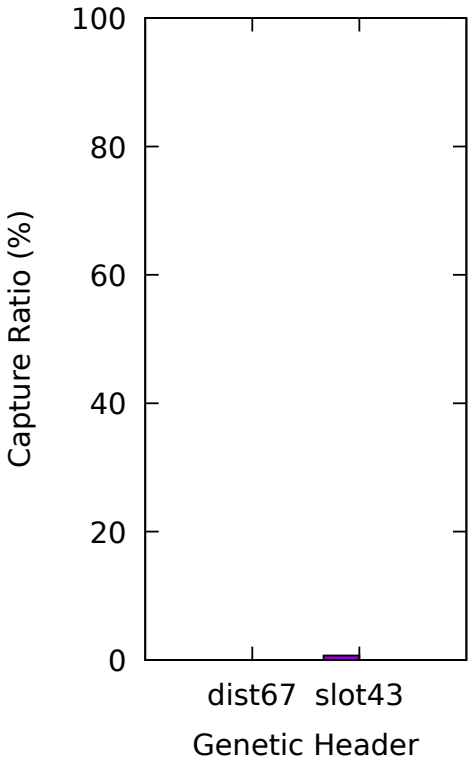


(d) Latency

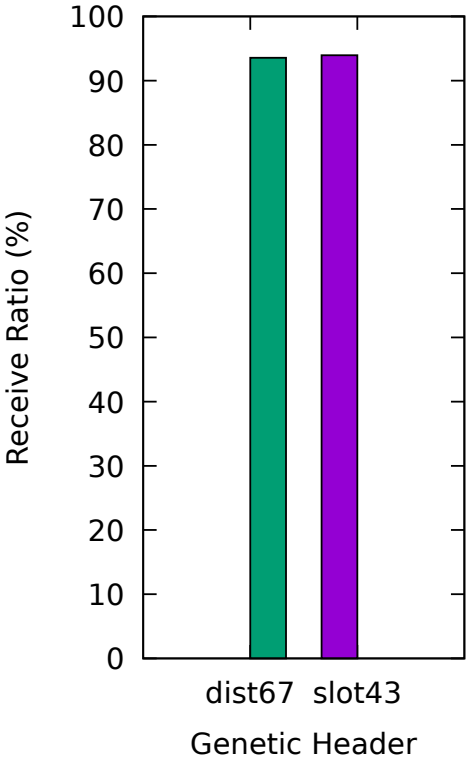
Figure 5.5: Simulation results for 100 outputs from the GA for different GA fitness functions and communication models



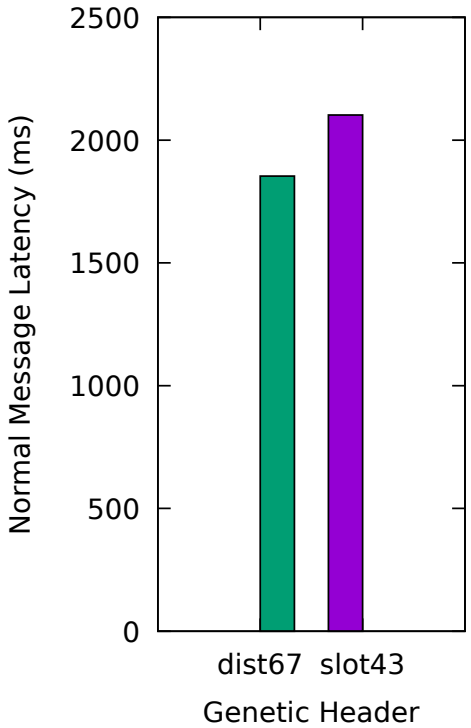
(a) Messages Sent



(b) Captured



(c) Receive Ratio



(d) Latency

Figure 5.6: Simulation results for the two solutions that lie on the Pareto frontier

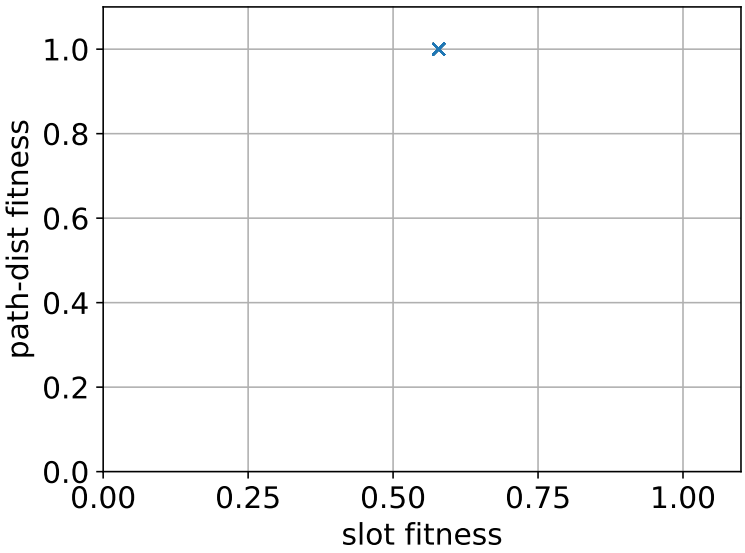
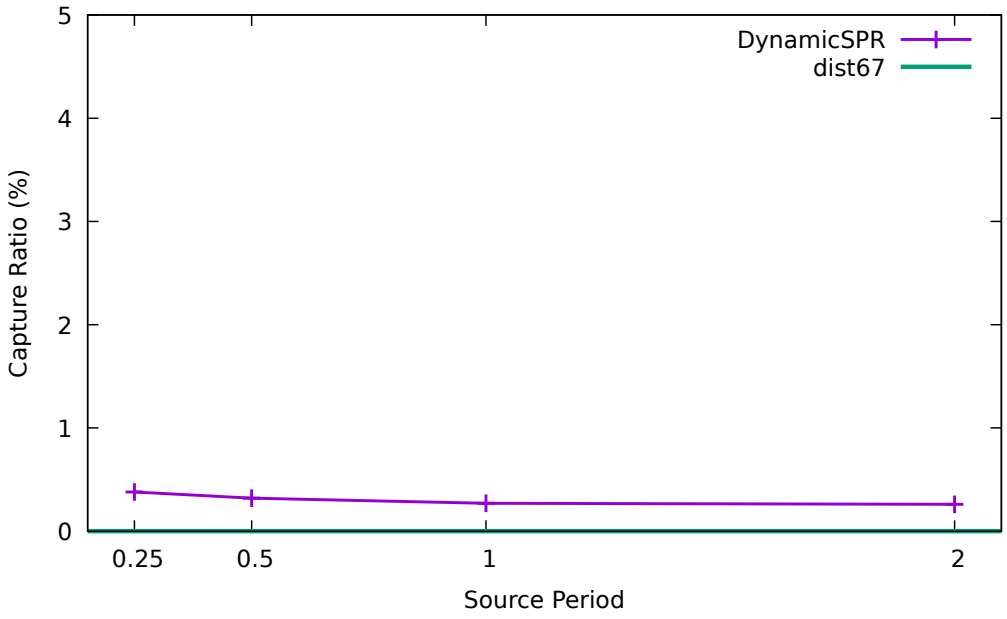
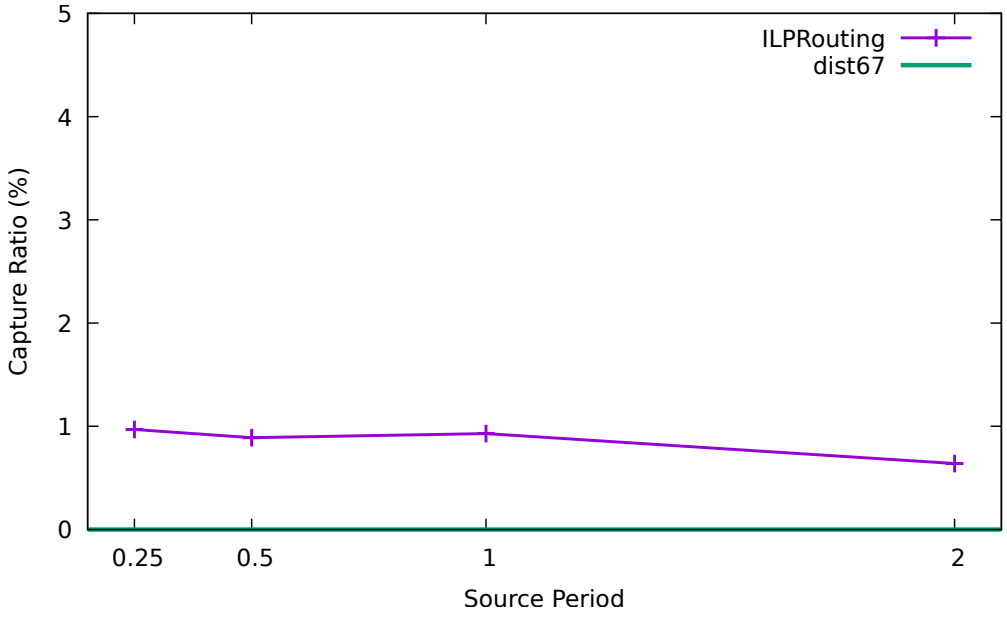


Figure 5.7: The Pareto frontier output by a single run of NSGA-II

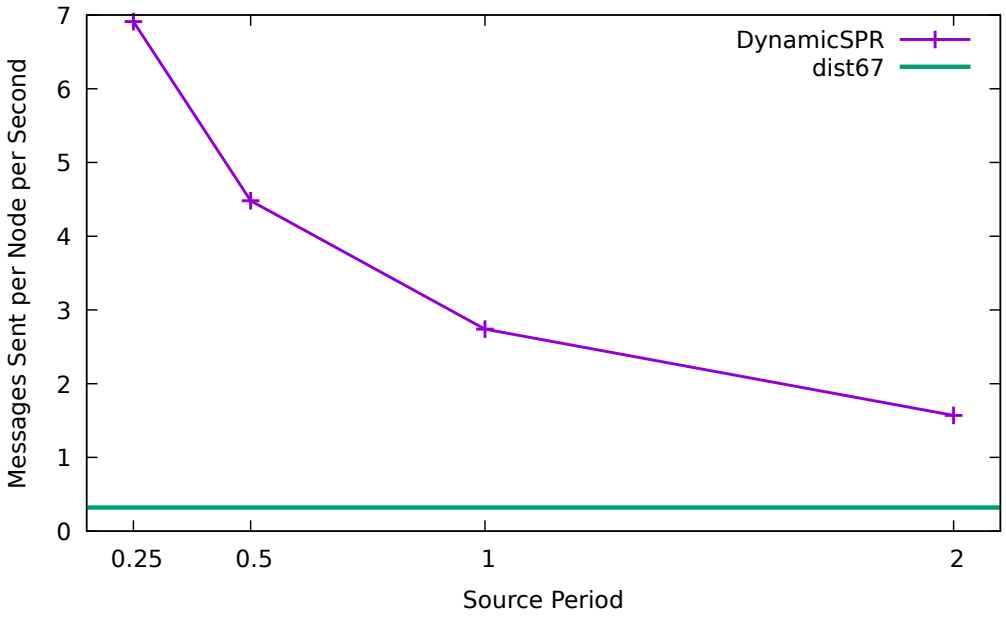


(a) DynamicSPR

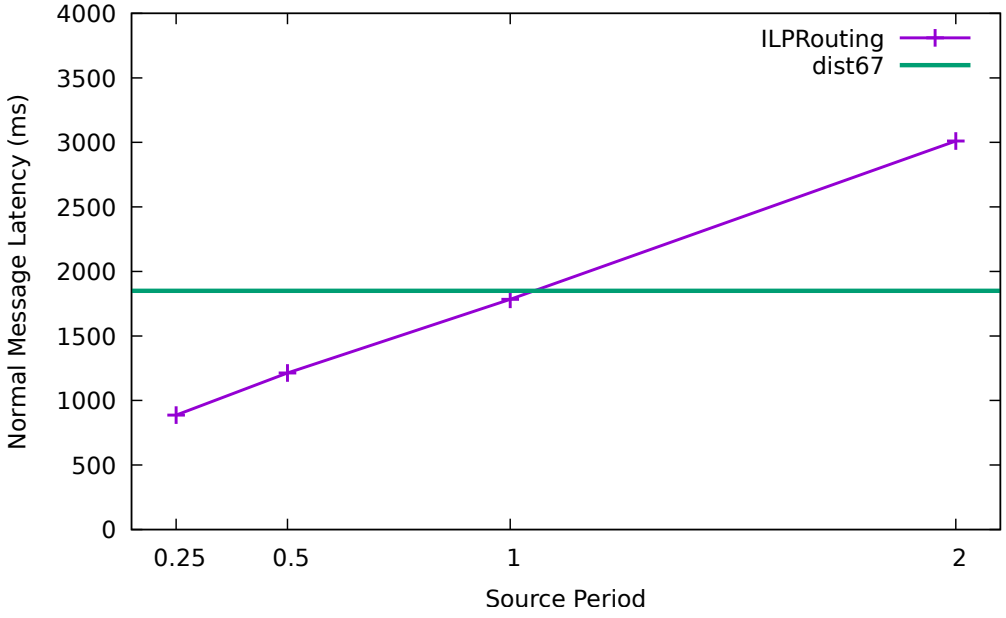


(b) ILPRouting

Figure 5.8: Capture ratios of different algorithms



(a) DynamicSPR messages sent



(b) ILPRouting latency

Figure 5.9: Various metrics from DynamicSPR and ILPRouting

Chapter 6

Providing Source Location Privacy Using an Online Distributed Algorithm

This chapter focusses on developing SLP-aware TDMA DAS schedules in an online, distributed fashion, in the presence of a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker as described in Section 3.2. This parametrisation is the principal model used in the vast majority of SLP works, and will be the primary method of analysing the solution defined in this chapter. As an addition to this parametrisation, further ones are analysed with respect to the final solution.

In this context, the term *online* is defined as performing all processing required for the solution to operate on the sensor nodes themselves. This does not exclude the use of a setup period for the network to establish itself, as long as said setup is performed on the nodes. Given this limitation, the solution will have significantly less information available to it than that of the offline solution presented in the previous chapter, including having no prior knowledge of the network topology beyond the existence of the sink node and no indication as to the location of the source node.

The purpose of providing an online solution is that it is highly likely that the exact topology of the network will not be known before deployment (as the information created by the offline solution was included in the binary to be flashed to the motes) and the location of the asset (and as such the source) is almost impossible to determine before the network is operational. Providing an online algorithm endeavours to make the concepts presented in this work so far more applicable to a real-world scenario.

6.1 Impossibility Results

In this section, some important results regarding SLP-awareness of DAS are presented.

Theorem 1 ($(1, 0, 1, s_0, D)$-attacker and Strong SLP DAS)**.** *Given a network $G = (V, E)$, a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, a safety period δ, a source $\mathcal{S}$, then there is no strong DAS slot assignment $\mathcal{F}$ such that $\mathcal{F}$ is δ-SLP-aware for $\mathcal{A}$ in the presence of $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker.*

Proof. Assume such an $\mathcal{F}$ exists. Since $\mathcal{F}$ is SLP-aware, then $\delta_{F,A} < \delta$. Denote the protectionless strong DAS slot assignment by $\mathcal{F}^s$. Now, assume that $\mathcal{S}$ is captured through the following path $pc^s = \langle s_0 s_1 \dots s_i \dots s_k (= \mathcal{S}) \dots \rangle$ (within δ_s) in $\mathcal{F}^s$ and also assume $(k - (i - 1)) = (\delta - \delta_s)$.

Now, assume $\mathcal{F}$ has a the same slot assignment as $\mathcal{F}^s$ for the first i locations of pc^s, i.e., $\forall j, 0 \leq j < i$, $\mathcal{F}(s_j) = \mathcal{F}^s(s_j)$. Since $\mathcal{F}$ is SLP-aware, then the attacker trace pc^s is not valid under $\mathcal{F}$. Since $\mathcal{F}$ and $\mathcal{F}^s$ have the same assignment for $s_0, \dots, s_{i-1}$, then, denote by pc the attacker trace under $\mathcal{F}$. Then, $pc = \langle s_0 \dots s_{i-1} s'_i \dots \rangle$ and assume $(s_i, s'_i) \in E$. This means that s'_i is the "first" node where the attacker needs to start diverging from pc^s to avoid capturing $\mathcal{S}$.

There are three cases to consider:

1. s_i and s'_i are the same distance from the sink. Since $\mathcal{F}$ is a strong DAS, then $\mathcal{F}(s'_i) < \mathcal{F}(s_i)$. Then, a path containing $\langle s'_i s_i \dots \text{sink} \rangle$ is not the shortest path, hence $\mathcal{F}$ cannot be a strong DAS.

2. s'_i is closer to the sink than s_i. Then, since $\mathcal{F}(s'_i) < \mathcal{F}(s_i)$, $\mathcal{F}(s'_i) < \mathcal{F}(s_i)$ cannot be a strong DAS, as it violates the DAS property.

3. s'_i is further from the sink than s_i. Then, $\mathcal{A}$ is one step closer to the source, meaning that $\mathcal{F}$ is $(\delta - 1)$-SLP-aware.

All three lead to a contradiction, so no such $\mathcal{F}$ exists. □

Theorem 1 suggests that it is impossible to develop a strong SLP-aware DAS. Thus, weak SLP DAS is the focus from this point onwards. Now, the aim is to determine whether it is possible to develop a weak SLP DAS that can provide SLP when any node in the network can be source.

Theorem 2 (Weak SLP for all nodes)**.** *Given a network $G = (V, E)$, a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, a safety period δ, then there is no weak DAS algorithm $\mathcal{F}$ such that $\mathcal{F}$ is δ-SLP-aware for every node $n \in V$ in the presence of $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker.*

Proof. Since $\mathcal{F}$ is SLP-aware for all $n \in V$, then it means that $\mathcal{A}$ cannot capture n within δ. At a given node n_i, the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker hears a message under $\mathcal{F}$ from node n_{i+1}, say. As R is 1, then the attacker has to move. Since $\mathcal{A}$ does not keep track of history, i.e., $H = 0$, $\mathcal{A}$ moves according to D, defined above. It then cannot move within this period again as $M = 1$. Now, assume the sequence of location visited by $\mathcal{A}$ within δ is $\mathcal{T} = \langle s_0 \cdot s_1 \dots s_k \rangle$. Since $\mathcal{F}$ is δ-SLP-aware for all $n \in V$, it means $n \notin \mathcal{T}$. However, there exists a node $s_i, 0 \leq i \leq k$ such that $s_i \in \mathcal{T}$, which is a contradiction. Hence, no such $\mathcal{F}$ can exist. □

The consequence of Theorem 2 is that a given DAS algorithm $\mathcal{F}$ can only provide SLP for some subset of nodes $\mathcal{N}^{\mathcal{F}}$. In other words, if a node $n \notin \mathcal{N}^{\mathcal{F}}$ is a source, then a new DAS schedule $\mathcal{F}'$ needs to be generated for n. As such, in the rest of the chapter, focus is on the set of nodes that are at least a certain distance ω from the sink. Typically, ω should be at least half the source-sink distance, i.e., the source should not be close to the sink. This set of nodes is denoted by G^{ω}.

Corollary 1. *Given a network $G^{\omega} = (V, E)$, a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, a safety period δ, then existing DAS algorithms $\mathcal{F}$ are not δ-SLP-aware for every node $n \in G^{\omega}$ in the presence of $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, for $\delta = \lambda$ rounds, where λ represents the minimum hop distance between a source and the sink.*

Thus, new DAS algorithms are needed to provide SLP to a set of nodes.

Typically, an existing DAS algorithm $\mathcal{F}$, for a given network $G = (V, E)$, can only provide δ-SLP to a subset of nodes $V^{\omega} \subset V$, where δ is equal to the (sink source distance) in time periods. However, as mentioned previously, a safety period $\delta^{G}_{\mathcal{S},\mathcal{A}} > \delta^{G}_{\mathcal{F},\mathcal{A}}$ is specified and the capture probability is evaluated of a SLP-aware DAS protocol $\mathcal{F}^{s}$ that is obtained by extending $\mathcal{F}$. In the next section, a novel *3-stage distributed protocol* is presented that achieves the transformation of $\mathcal{F}$ into $\mathcal{F}^{s}$.

6.2 Algorithms

The 3-stage protocol works as follows: Firstly, a *normal* DAS schedule is generated. Then, a suitable location is searched for in the network where the attacker can be *tricked* for some time. Finally, the *trick* is to reassign slots to some nodes to ensure that the attacker takes a longer route towards the source, thus delaying it (causing the safety period to expire). While slots are reallocated in stage 3, an important property of the distributed protocol is that the DAS property is preserved.

Algorithm 8 shows a protocol that generates a DAS schedule, Algorithm 9 searches for a suitable location in the network to start a redirection and the final

phase is shown in Algorithm 10. The working of the protocols will now be explained and their correctness shall be proven.

6.2.1 Phase 1: DAS Schedule

The protocol for the DAS schedule (Algorithm 8) is started by the sink. A node keeps track of its potential parents, its distance from the sink and its allocated slot. When it receives a set of "non-empty" messages, it chooses one of the senders as its parent and also updates its slot to be less than the minimum of all slots seen. This novel aspect is to ensure that, when the slot assignment is subsequently refined, then the sender can very easily identify a new parent. This ensures the DAS property is satisfied.

On the other hand, when a node broadcasts its state, it includes the state of its neighbours. A receiving node will then potentially have information about its 2-hop neighbours. It then verifies whether it shares the same slot as one of its 2-hop neighbours. If it does, then one of the two colliding neighbours will update its slot value, guaranteeing collision-freedom.

The algorithm satisfies the DAS property as follows:

Lemma 1 (DAS property)**.** *Given a network $G = (V, E)$, a sink $s \in V$, the following invariant holds for the algorithm of Algorithm 8*

$$\forall j, k$$

$$I1 \;\; j.hop = \bot \vee j.par = \bot \equiv j.slot = \bot \wedge j.Normal = 1$$

$$I2 \;\; j.slot \neq \bot \wedge j.par = k \Rightarrow k.slot > j.slot$$

$$I3 \;\; j.slot = k.slot \wedge j.Ninfo[k] \neq \bot \Rightarrow (j.slot' \neq j.slot \vee k.slot' \neq k.slot)$$

Proof. $I1$ follows trivially Algorithm 8. $I2$ is achieved by action *process*. $I3$ is also achieved by the collision resolution part of the *process* action □

Lemma 1 shows that the protocol of Algorithm 8 satisfies both the DAS property and the collision freedom property.

6.2.2 Phase 2: Node Locator

The node locator protocol of Algorithm 9 searches for a node which is at a certain distance from the sink. The idea is that the distance of that node is far enough from the source, typically roughly $\omega/2$, and is shown through the following lemmas.

Algorithm 8 Phase 1 - DAS algorithm: A slot assignment protocol for data aggregation scheduling.

▷ $\mathcal{F}$: DAS protocol for process i
1: **variables**
2: myN: set of int ▷ Set of neighbours
3: $Npar$: set of int **init** $\emptyset$ ▷ Set of potential parents
4: $children$: set of int **init** $\emptyset$ ▷ Set of children
5: $Others[]$: array of set of int **init** $\emptyset$ ▷ Set of potential competitors
6: $Ninfo[]$: array of (int,int) ▷ 2-hop neighbourhood information (hop,slot)
7: $hop, par, slot$: int,int,int **init** $\perp, \perp, \perp$ ▷ DAS information
8: $Normal$: bool **init** 1 ▷ Flag for re-computation of DAS
9: $start$: bool **init** 1 ▷ Flag to begin algorithm
10: $dissem$: timer **init** α ▷ Timer for broadcasting neighbourhood information

11: **constants**
12: $sink$: bool ▷ 1 if node is sink, otherwise 0
13: $length$: int ▷ Length of redirection
14: $size$: int ▷ Size of the network

15: $init :: start \rightarrow$ ▷ Sink triggers the protocol
16: **if** $sink$ **then**
17: **for all** $n \in myN$ **do** $Ninfo[n] \leftarrow (\perp, \perp)$
18: $hop, par, slot, start \leftarrow 0, \perp, size, 0$
19: $Ninfo[i].hop, Ninfo[i].slot \leftarrow hop, slot$

20: $dissem ::$ **timeout**$(dissem) \rightarrow$ ▷ When $dissem$ times out
21: **if** $slot \neq \perp$ **then**
22: **broadcast** DISSEM$\langle Normal, \{Ninfo[j] \mid j \in myN\}, par\rangle$
23: **set**$\langle dissem, \alpha\rangle$

▷ Normal dissem message received
24: $receiveN ::$ **receive** DISSEM$\langle 1, N, p\rangle$ **from** $j \rightarrow$
25: **if** $slot = \perp \wedge N[j].slot \neq \perp$ **then**
26: $Npar \leftarrow Npar \cup \{j\}$
27: $Others[j] \leftarrow Others[j] \cup \{n | N[n].slot = \perp\}$
28: **for all** $n \in myN \mid N[n] \neq \perp$ **do** $Ninfo[n] \leftarrow N[n]$

▷ Non-normal (update) dissem message received
29: $receiveU ::$ **receive** DISSEM$\langle 0, N, p\rangle$ **from** $j \rightarrow$
30: **if** $par = j$ **then**
31: **if** $slot \geq N[j].slot$ **then**
32: $slot \leftarrow N[j].slot$
33: $Normal \leftarrow 0$
34: **for all** $n \in myN \mid N[n] \neq \perp$ **do** $Ninfo[n] \leftarrow N[n]$

Algorithm 8 (cont.) Phase 1 - DAS algorithm: A slot assignment protocol for data aggregation scheduling.

35: *process* :: **rcv**$\langle\rangle \rightarrow$ ▷ Finished receiving dissem messages
36: **if** $slot = \bot$ **then**
▷ Select parent and calculate DAS info
37: $hop \leftarrow min\{h \mid (h, s) \in Ninfo[k], k \in Npar\} + 1$
38: $par \leftarrow min\{k \mid k \in Npar, Ninfo[k] = (hop, s)\}$
39: $slot \leftarrow Ninfo[par].slot - rank(i, Others[par]) - 1$
40: $Ninfo[i] \leftarrow (hop, slot)$
41: **for all** $n \in myN \mid Ninfo[n].slot = \bot$ **do**
42: $children \leftarrow children \cup \{n\}$
▷ Detect and resolve collisions
43: **if** $\exists j, j \neq i : Ninfo[j].slot = slot$ **then**
44: **if** $hop > Ninfo[j].hop \vee (hop = Ninfo[j].hop \wedge i > j)$ **then**
45: $slot \leftarrow slot - 1$

Lemma 2 (Node locator)**.** *Given a network $G = (V, E)$, a sink s, and a distance pr, then the following is an invariant of Algorithm 9:*

$\forall j, k$

I1 $k.startNode \wedge k.par = j \Rightarrow (rank(k, \text{children[j]}) = |\text{children[j]}|) \wedge k.hop = pr$

I2 $k.par = j \wedge (rank(k, \text{children[j]}) = |\text{children[k]}|) \wedge j.hop < pr \Rightarrow BCAST\langle SEARCH, j, j.hop + 1, pr\rangle \in j.k.ch$

Proof. $I1$ is satisfied by the first branch of the *ReceiveS* action while $I2$ is satisfied by the second branch of *ReceiveS*. □

Lemma 3 ($(1, 0, 1, s_0, D)$–$\mathcal{A}$ attacker and node locator)**.** *Given a network $G = (V, E)$, a DAS assignment $\mathcal{F}$, and a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker and distance pr, then $\mathcal{A}$ will reach the start node in pr rounds.*

Proof. Under the assumption of a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, $\mathcal{A}$ can receive a single message before going to the sender of that message, i.e., the only possible definition for D. In the first round (i.e., first period), $\mathcal{A}$ will move from the sink to the node from which it first receives a message, i.e., it will move to the child with the *lowest* slot among s_0's children. Denoting the number of children the sink has by y, $\mathcal{A}$ will move to the child node i with slot $\Delta - y - 1$ as a consequence of this condition: (rank(i, $Others[par]$) = $|Others[par]|$). Because $\mathcal{A}$ moves to a child node with a smaller slot value, it has to wait for the next period. Thus, the same process is repeated in subsequent periods until a node with hop value pr is reached by $\mathcal{A}$, which the attacker reaches after pr periods. □

Algorithm 9 Phase 2 - Node Locator: An algorithm that searches for a suitable location in the network for where the redirection can occur. The algorithm inherits variables from Algorithm 8.

```
    ▷ Search protocol for process i
 1: variables
 2:   startSearch: bool init 1                          ▷ Start of search
 3:   startNode: bool init 0                            ▷ First node of redirection
 4:   pr: int init ⊥                                    ▷ Length of redirection
 5:   from: set of int init ∅                           ▷ To prevent backtracking

 6: constants
 7:   dist: int init SD                                 ▷ Length of search path
 8:   diam: int init δ                                  ▷ Diameter of network

 9: startS :: startSearch ∧ sink →                      ▷ Begin search process
10:   for all c ∈ children do
11:     if Ninfo[c].slot = min{Ninfo[k].slot | k ∈ children} then
12:       startSearch ← 0
13:       pathChild ← c
14:   broadcast SEARCH⟨pathChild, dist⟩

    ▷ Receive search message
15: receiveS :: receive SEARCH⟨k, d⟩ from j →
16:   from ← from ∪ {j}
17:   if i = k then
18:     if (d = 0) ∧ (Npar \ {par, j} ≠ ∅) then
19:       startNode ← 1
20:       pr ← ⌈dist/3⌉
21:     if (d = 0) ∧ (Npar \ {par, j} = ∅) then
22:       if children ≠ ∅ then
23:         pathChild ← choose(children)
24:       else
25:         pathChild ← choose(myN \ {par})
26:       broadcast SEARCH⟨pathChild, d⟩
27:     if d > 0 then
28:       for all c ∈ children do
29:         if Ninfo[c].slot = min{Ninfo[k].slot | n ∈ children} then
30:           pathChild ← c
31:       broadcast SEARCH⟨pathChild, d − 1⟩
```

6.2.3 Phase 3: Slot Refinement

The protocol of Algorithm 10 works as follows: when a node is selected from the node selector algorithm of Algorithm 9, the selected node becomes the first to trigger a change in slots of some nodes. The number of nodes to change slots is either determined by $lenD$ when nodes have at least two potential parents or until it encounters a node with only one potential parent. The selected node m chooses one of its potential parents (but not its parent) as the node to change its slot, i.e., an "upstream" node n will change its slot. For the attacker to move to n first, the slot value of n needs to be smaller than all the other nodes in m's neighbourhood. When n changes its slot, it has to inform its children to update their slots. This is achieved by setting *Normal* to 0, i.e., it is an update phase.

Now, in the DAS algorithm (Algorithm 8), whenever a node disseminates its state information, it specifies whether the dissemination is an update or not. If it is an update, a node checks if its slot value requires changing. If it does, then it changes its slot value to a value less than its parent (to maintain the DAS property) and also informs its children that an update is required. Up to $lenD$ nodes will start a slot update.

Lemma 4 (DAS preservation)**.** *Given a network $G = (V, E)$, a sink s_0, a DAS slot assignment $\mathcal{F}$ (Algorithm 8), a start node n at a distance pr from s_0 and a redirection distance λ, then ($\mathcal{F}^s = \mathcal{F} \circ SRefine$) satisfies the DAS property for G.*

Here, $A \circ B$ means executing slot assignment A followed by that of B.

Proof. This is proven in two parts. For the first part, the nodes that have not changed their slots during the refinement are examined and then nodes that were updated in the second part.

First part: A node n that hasn't changed its slot value has not changed because it either did not receive an *update* message from its parent m, i.e., a message with *Normal* set to 0, or its slot is still smaller than its parent. This is trivially satisfied in the DAS protocol through the action *receiveU*.

Second part: When a node n updates its slot, all of its children will receive an update message through *receiveU* action. Its children will update their slot by decreasing their slot values by an amount identical to that of n. Thus, all the children of n will have slot values less than that of n, preserving the DAS property. Also, a node n may update its slot because it received a *CHANGE* message. Since $\mathcal{F}$ was a DAS protocol, and as the refinement then decreases its slot by 1 in action *receiveC*, DAS is preserved. □

Lemma 5 (Collision freedom preservation)**.** *Given a network $G = (V, E)$, a sink s_0, a DAS slot assignment $\mathcal{F}$ (Algorithm 8), a start node n at a distance pr from s_0 and a redirection distance λ, then the slot assignment ($\mathcal{F}^s = \mathcal{F} \circ SRefine$) satisfies the collision-freedom property for G.*

Proof. As $\mathcal{F}$ is collision-free and, when a node n decreases its slot by a given value λ, all of its children also decreases their respective values by λ in action *receiveU*. Also, if the new slot now collides with some other nodes, then this is resolved by action *process*. Thus, collision-freedom is preserved. □

Lemma 6 (SLP provision)**.** *Given a network $G = (V, E)$, a DAS slot assignment ($\mathcal{F}^s = \mathcal{F} \circ$SRefine*$)$*, where $\mathcal{F}$ is a weak DAS assignment, a source $\mathcal{S}$, a search distance α, a redirection distance λ, a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, then $\mathcal{F}^s$ is a weak δ-SLP-aware DAS protocol for $\mathcal{S}$ in G in the presence of $\mathcal{A}$, where $\delta = max(\delta^G_{\mathcal{P},\mathcal{A}}, O(\alpha + \lambda))$*

Proof. $\mathcal{F}^s$ is a weak DAS follows from Lemmas 4 and 5. Also, since the search path is α, the node at the end of the search asks a potential parent to change its slot. This is repeated for λ nodes. Thus, the node at the end of the change chain will be at a position of at most of $\alpha + \lambda$ from the sink, requiring $(\alpha + \lambda)$ periods for the attacker to reach $\mathcal{S}$. On the other hand, if the attacker is drawn closer to $\mathcal{S}$ during redirection, this is no less than $\delta^G_{\mathcal{P},\mathcal{A}}$. □

Theorem 3 (SLP-awareness)**.** *Given a network $G = (V, E)$, a DAS slot assignment ($\mathcal{F}^s = \mathcal{F} \circ$*SRefine*$)$*, where $\mathcal{F}$ is a weak DAS assignment, a source $\mathcal{S}$, a search distance α, a redirection distance λ a $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker, then there exists a safety period $\delta^G_{\mathcal{S},\mathcal{A}}$ such that $\mathcal{F}^s$ is a weak $\delta^G_{\mathcal{S},\mathcal{A}}$-SLP-aware DAS protocol for $\mathcal{S}$ in G in the presence of $\mathcal{A}$.*

Proof. This follows directly from Lemma 6. □

The following is an important remark: The safety period is related to the network topology and the source location, i.e., different network topologies and source locations will have different capture times, hence safety periods. Thus, for a given safety period δ, attacker $\mathcal{A}$ and source $\mathcal{S}$, there can exist networks G, G' such that $\delta^G_{\mathcal{S},\mathcal{A}} < \delta^G_{\mathcal{F}^s,\mathcal{A}} < \delta^{G'}_{\mathcal{S},\mathcal{A}}$.

Algorithm 10 Phase 3 - Slot Refinement: An algorithm that refines the original slot assignment $\mathcal{F}$. The algorithm inherits variables from Algorithms 8 and 9.

▷ Slot refinement protocol for process i

1: $startR :: startNode \rightarrow$ ▷ Begin the change process
2: $startNode \leftarrow 0$
3: $pathChild \leftarrow choose(Npar \setminus \{par\} \setminus from)$
4: $nSlot \leftarrow \min(\{Ninfo[j].slot \mid j \in myN\} \cup \{slot\})$
5: **broadcast** CHANGE$\langle pathChild, nSlot, pr - 1\rangle$

6: $receiveC$ :: **receive** CHANGE$\langle p, s, d\rangle$ **from** $j \rightarrow$
7: **if** $d > 0 \wedge i = p$ **then**
8: **if** $myN \setminus \{par\} \setminus from \neq \emptyset$ **then**
9: $pathChild \leftarrow choose(myN \setminus \{par\} \setminus from)$
10: $slot \leftarrow s - 1$
11: $Ninfo[i] \leftarrow (hop, slot)$
12: $nSlot \leftarrow \min(\{Ninfo[k].slot \mid k \in myN\} \cup \{slot\})$
13: **broadcast** CHANGE$\langle pathChild, nSlot, d - 1\rangle$
14: **if** $d = 0 \wedge i = p \wedge (myN \setminus \{par\} \setminus from \neq \emptyset)$ **then**
15: $Normal, slot \leftarrow 0, s - 1$
16: $Ninfo[i] \leftarrow (hop, slot)$

6.3 Experimental Setup

6.3.1 Algorithm Parameters

Table 6.1 provides a list of the parameters used by the algorithms in these simulations. SLP DAS inherits all of the parameters from protectionless DAS and adds two additional parameters (search distance and change length). The search distance is the number of hops search messages travel away from the sink in order to discover a suitable starting point for the change path. The change length is the maximum length of the path the attacker will follow created by altering the slot assignments of each node within the path. The change length parameter is set to be the sink-source distance minus the search distance to avoid the case where the attacker ends up close to the source after its redirection[1]. This makes the search distance plus the change length equal to the sink-source distance.

The protectionless DAS algorithm does not have a safety period but in order to create a capture ratio the simulations have an upper bound of *number of nodes* $\times$ *source period* $\times 4$, beginning from the point where the first normal message is sent. This is because if the attacker does not capture the source before this point then most likely the attacker has been led away from the source and is trapped in some other area of the network (i.e. one of the corners). An upper bound is required for the simulation time mainly as an implementation requirement. Without an upper bound, should the attacker get trapped in the network looping between a set of nodes, the simulation would never terminate. The reason that this particular value was selected is because if an attacker performs an exhaustive search of the entire network, it will have completed before the upper bound expires even if a significantly sub-optimal method of search was chosen. Should the upper bound be reached, the conclusion is drawn that the attacker will never find the source and an exhaustive search would have been a better option (but not viable as there is an implicit assumption that the asset and source will have relocated before an exhaustive search of the network can be completed).

If the attacker is to capture the source, the expected time from when the first normal message is sent is *period length* $\times (\Delta(Sink - Source) + 1)$, as this is the time it would take to traverse one of the shortest paths to the source. The safety period for SLP DAS was calculated as *period length* $\times (\Delta(Sink - Source) + 1) \times 1.5$, where 1.5 is the factor of additional time that SLP DAS is expected to provide over protectionless DAS.

The routing algorithm being used by both protectionless DAS and SLP DAS is flooding. Each node broadcasts a message in its time slot, regardless of whether it

[1]Please observe that this can still happen depending on the specific network topology.

has received a message from the source. This is because SLP DAS relies on periodic message broadcasts in order to function correctly, as without them there would only be messages sent by the source which would cause the attacker to follow one of the shortest paths directly to the source.

Parameter	Description	Value
	Protectionless DAS	
Source Period	The rate at which the source generates messages	15s
Slot Period	The duration of a single slot	0.1s
Dissemination Period	The duration of the dissemination period	5s
Number of Slots	The number of slots that can be assigned	100
Minimum Setup Periods	The number of periods before the source is activated	60
Neighbour Discovery Periods	The number of periods for neighbour discovery	4
	SLP DAS	
Search Distance	The maximum number of hops search messages make	3, 5, 7
Change Length	The length of the change path generated	$\Delta(Sink - Source) - SearchDistance$

Table 6.1: A list of parameters for the protectionless and SLP DAS algorithms

6.3.2 Attacker Model

The attacker model is implemented as described in Section 3.2, with an implementation satisfying the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker for the initial investigation. It is assumed that the attacker is aware of the period length and thus only performs one move per period.

Further experiments compare different parametrisations of the (R, H, M, s_0, D)-$\mathcal{A}$ attacker model.

6.4 Results

6.4.1 Expected Outcomes

The metrics that shall be examined for the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ to determine the efficiency of the resulting SLP DAS algorithm are (i) capture ratio, (ii) messages send rate per node, (iii) received ratio and (iv) latency. The differences between protectionless DAS and SLP DAS will be compared with varying values for search distance.

It is expected that the capture ratio of SLP DAS for any search distance to be lower than its protectionless counterpart. This is not to say that the expectation is a 0% capture ratio, as SLP DAS does not take into account the topology of the network and as such will sometimes create a path to the source. Comparing the ideal communication model to low-asymmetry, the capture ratio depends on the number of message collisions and lost messages but as there will be more for low-asymmetry it is expected that a lower capture ratio for both algorithms using that model.

The message send rate per node should be similar for both algorithms and regardless of search distance. While the SLP algorithm will have the additional overhead of search and change messages, this is negligible.

The received ratio depends on the communication model, with the ideal model approaching 100% and the low-asymmetry model being lower but similar for all algorithms.

Finally, the normal message latency is expected to be bounded at a maximum of the slot period multiplied by the number of slots, as this is the maximum time it would take to travel from the point furthest from the sink to the sink in a period. Additionally, the latency is expected to increase as the network gets larger, as the distance between the sink and the furthest point in the network increases. The latency of both communication models should be similar, as lost messages are discounted from the latency calculation.

Further investigation will be performed into the effectiveness of varying attacker models. A comparison is made between the models: $(1, 0, 1, s_0, D)$-$\mathcal{A}$,

$(1, 0, 2, s_0, D)$-$\mathcal{A}$, $(2, 0, 1, s_0, D)$-$\mathcal{A}$, $(2, 0, 2, s_0, D)$-$\mathcal{A}$, $(2, 1, 2, s_0, D)$-$\mathcal{A}$ and $(2, 2, 2, s_0, D)$-$\mathcal{A}$. For this examination, 5 is selected as the search distance parameter and run SLP DAS with the stated attacker model parametrisations.

When the network is reliable, it can be expected that the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker will be strongest attacker as it will receive a message from a node closer to the source first and it will make the move to that node. Specifically, the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker will not make a move that is worse than other attackers. On the other hand, when the network is unreliable, e.g., when links are unidirectional or links are asymmetric, then the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ may not receive a message from a node closer to the source first. This then reduces the efficacy of the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker. On the other hand, attackers that can make multiple moves in a given period tend to be more powerful, as they can potentially overcome the problem of unreliable links.

6.4.2 $(1, 0, 1, s_0, D)$-$\mathcal{A}$ Attacker

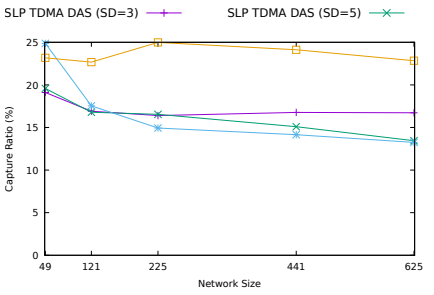
In this section, the results are presented showing the differences between protectionless and SLP DAS, and compare them to the expected outcomes that were presented in Subsection 6.4.1.

The capture ratio (shown in Figure 6.1) follows the intuition of the expected outcome, with each SLP DAS search distance parametrisation providing a lower capture ratio than protectionless DAS.

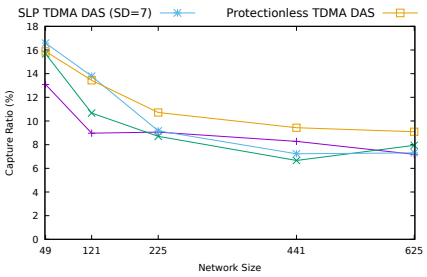
The message send rate (shown in Figure 6.2) does not follow the expectation, as protectionless DAS appears to send messages more frequently than the SLP DAS algorithms. This counter-intuitive result is due to protectionless DAS sending more dissemination messages in the setup phase than SLP DAS. This is because two periods in the SLP DAS setup phase are reserved for sending search and change messages only, where protectionless DAS has every node send dissemination messages in these periods.

Receive ratio (shown in Figure 6.3) is also as expected. Each algorithm is almost identical, with the ideal communication model having a near 100% receive ratio and the low-asymmetry communication model delivering less messages due to noisy conditions.

Finally, normal message latency (shown in Figure 6.4) is also as expected, giving near-identical measurements across algorithms and communication models. As stated previously, these plots show that this latency is indeed bounded by the number of slots multiplied by slot length (which totals 10s).

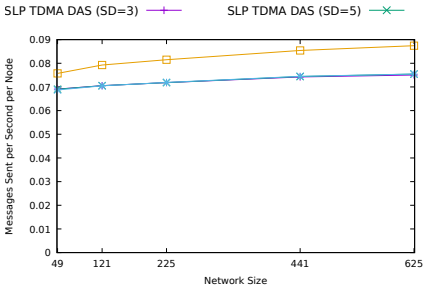


(a) Ideal communication model

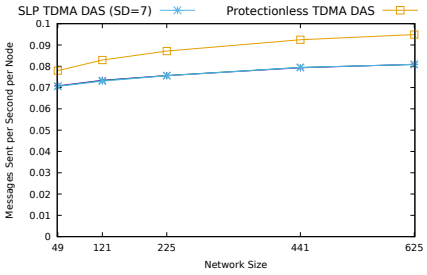


(b) Low asymmetry communication model

Figure 6.1: Capture ratio

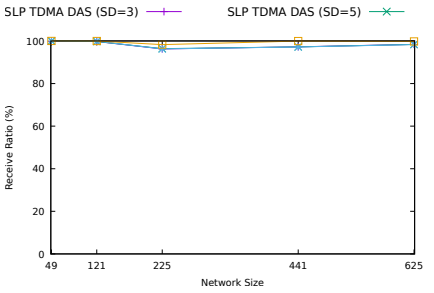


(a) Ideal communication model

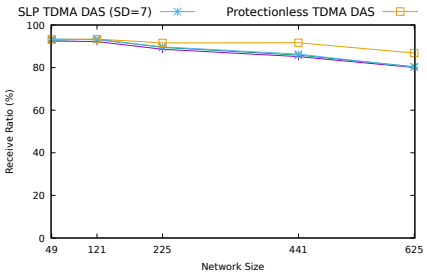


(b) Low asymmetry communication model

Figure 6.2: Messages sent per node per second



(a) Ideal communication model



(b) Low asymmetry communication model

Figure 6.3: Received ratio

6.4.3 Further Attacker Model Parametrisations

In this section, the results showing the differences between attacker model parametrisations for protectionless DAS and SLP DAS (search distance of 5) are presented, and compared to the expected outcome presented in Subsection 6.4.1, which is that the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker is the strongest of the selected models.

The only metric will shall examine is capture ratio as the attacker model used has no ability to affect any of the other metrics, as the model cannot alter a node's behaviour or interfere with network traffic. The capture ratios for each attacker model are shown in Figure 6.5. As can observed, the highest capture ratio under the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker suggests that this attacker model is indeed the strongest attacker when the network is reliable (Figure 6.5a and Figure 6.5b). This is due to the reasons explained earlier in Subsection 6.4.1. On the other hand, attackers that can make more than one move in a single period are shown to be strong when the network is unreliable, as they have higher capture ratios for protectionless DAS. The efficiency of SLP DAS is shown, as the capture ratios of all attackers is reduced. However, attackers that can make a single move per period, such as $(1, 0, 1, s_0, D)$-$\mathcal{A}$ or $(2, 0, 1, s_0, D)$-$\mathcal{A}$ will have their capture ratio relatively unchanged in an unreliable network due to the non-determinism in the move they make, i.e., they may never take the diversionary path created.

6.4.4 Comparison with other SLP solutions

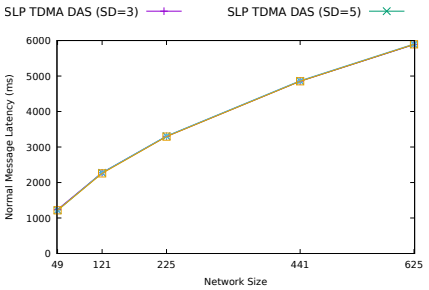
Compared to the offline solution presented in the previous chapter, this solution pales in comparison. However, this is largely to be expected as the offline solution had a perfect, global view of the network while this solution has no such information, making a comparison between the two nonsensical.

Should any other SLP protocols arise that operate at the MAC-layer of the network stack, a decent comparison could be made. However, comparing this solution to those that operate at the routing layer, such as ILPRouting and DynamicSPR, where a full view of the network topology can potentially be collected, is not a reasonable comparison to make and doesn't provide any real insight into the abilities of this solution.

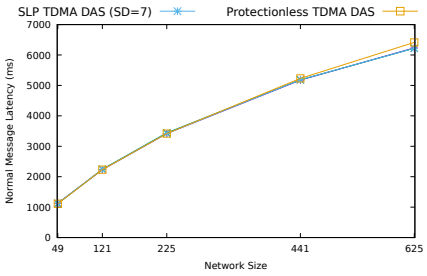
6.5 Conclusion

The objective of this chapter is to investigate the novel problem of developing a TDMA DAS schedule that can provide SLP. A number of impossibility results have been proven and, to counter these negative results, a 3-stage distributed algorithm

has been developed that transforms any DAS protocol into a corresponding SLP-aware DAS to a specific class of eavesdroppers. An advantage of SLP-aware DAS is that traffic re-engineering occurs at no message cost while reducing the capture ratio.

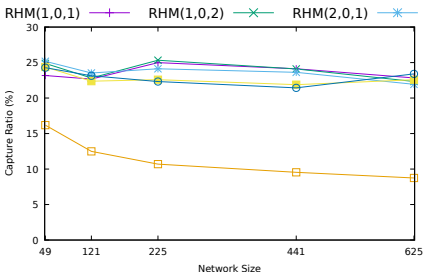


(a) Ideal communication model

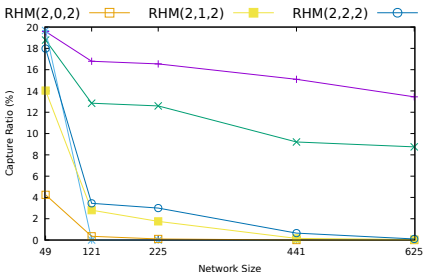


(b) Low asymmetry communication model

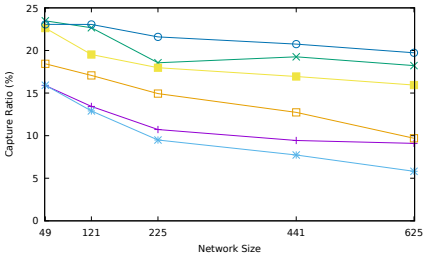
Figure 6.4: Normal latency



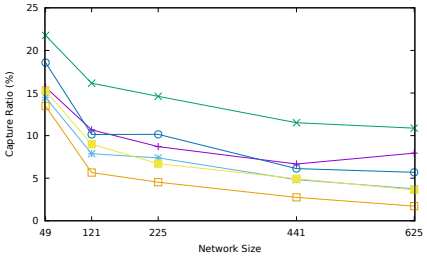
(a) Protectionless TDMA DAS — Ideal communication model



(b) SLP TDMA DAS — Ideal communication model



(c) Protectionless TDMA DAS — Low asymmetry communication model



(d) SLP TDMA DAS — Low asymmetry communication model

Figure 6.5: Capture ratio

Chapter 7

Providing Fault Tolerance and Source Location Privacy

One of the main challenges to overcome in WSNs is node crashes, which occur when nodes run out of energy or succumb to hardware or software defects. Spatial and temporal redundancy is necessary and sufficient to design for fault tolerance. A node crash will cause a path that can possibly lead the attacker away from the source to disappear and the new path followed by the attacker may allow them to reach the source.

Many works exist covering fault tolerance of a variety of aspects in WSNs, e.g., [78] presents an algorithm for implementing a crash-tolerant version of DAS. However, the technique shown in Chapter 6 is not tolerant to node crashes. There exists no work in the domain of SLP that deals with such faults.

In this chapter, a modified version of the algorithm presented in the previous chapter is presented. The algorithm provides spatial redundancy by creating *backup* diversionary routes to keep the attacker away from the source. First, the fault-tolerant DAS problem is shown to be NP-complete. Secondly, an eventually weaker version of fault-tolerant DAS is designed, called *eventually fault-tolerant* SLP-aware DAS. Finally, the distributed algorithm is shown, adhering to the eventually fault-tolerant SLP-aware DAS specification. This algorithm works in networks where nodes have at least two potential parents, i.e. there are at least two different paths between every node and the sink.

7.1 Complexity of Design of Fault-Tolerant SLP-aware (FT-SLP) Weak DAS

In this section, analysis of the complexity of developing a fault-tolerant SLP-aware weak DAS schedule is performed (as strong DAS is impossible to design). The objective is to develop a schedule that can tolerate f node crashes while retaining the SLP-awareness property. The challenge is that, when a node on an attacker path crashes, the attacker will follow a different path, which may lead to capturing the source. For simplicity, it is assumed that all the crashes occur on neighbours of the sink.

Multipath routing [5] is a well-known technique for providing fault tolerance. To preserve the SLP property when a node crashes, the attacker has to be "provided" with a route that takes them away from the source. Specifically, if up to f node crashes can occur, then there should exist at least f node-disjoint paths that lead the attacker away from the source. Having multiple node-disjoint paths is more achievable than one might think. [87] is an example of a testbed which has a very high density of nodes resulting in many disjoint paths. A path needs to be of such length that, once the attacker reaches the end of that *diversionary* path, he has no chance to capture the source within the safety period even if he attempts to trace back to the source. Node disjoint paths are needed so that no two paths are affected by a single crash. Thus, FT-SLP DAS can be designed through the development of multiple disjoint attacker paths. To simplify the following explanation, it is assumed that all crashes occur to nodes that are neighbours of the sink. The problem of finding node disjoint attacker paths is now formalised and it is shown that the problem is intractable.

Definition 12 (Disjoint Attacker Paths (DAP))**.** *Given a network $W = (N, L)$, a sink V_Δ, a length l, a number f, and a weak DAS/convergecast schedule Ξ, is there a set $\mathcal{P} = \{P_1 \dots P_{f+1}\}$ of attacker paths under Ξ such that:*

1. $\forall i, 1 \leq i \leq f+1 : length(P_i) = l$
2. $P_1 \dots P_{f+1}$ *are node-disjoint.*
3. $\forall i, 1 \leq i \leq f+1 : P_i$ *is not a capture path.*

The length l in the DAP problem is the minimum length of the *diversionary* path. The greater the l, the longer the path but also the path may end at a node close to the source. So, for simplicity, the paths are required to be of length l. The sequence of vertices on path are denoted P_i by $P_i = \langle n_i^1 \dots n_i^l \rangle$. The first result follows.

Lemma 7 (DAP & class of NP)**.** *DAP is in NP.*

Proof. To prove this, the correctness of $\mathcal{P}$ needs to be verified in polynomial time. So, given an instance of DAP and a solution set $\mathcal{P}$, the correctness of $\mathcal{P}$ is verified as follows:

1. The first condition is trivially verified.
2. The second condition can be verified in $O(f^2)$ through set operations.
3. The third condition can be verified in $O(f \cdot l)$ through enumeration.

□

The next result captures the complexity of designing FT-SLP.

Lemma 8 (DAP & NP-hardness)**.** *DAP is NP-hard.*

To prove this, firstly, the definition of the *disjoint connecting paths* problem is provided.

Definition 13 (Disjoint Connecting Paths (DCP))**.** *Given a graph $G = (V, E)$ and a set $T = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k pairs of vertices, does there exist k vertex-disjoint paths $P_1, P_2, \ldots, P_k$ such that P_i connects s_i to t_i?*

Proof. DAP can be reduced to DCP, first through the following mapping:

1. $W \mapsto G$
2. $f \mapsto k$
3. $n_i^1 \mapsto s_i, n_i^l \mapsto t_i, \forall 1 \leq i \leq f + 1$
4. $l \mapsto \min\{length(P_i), 1 \leq i \leq k\}$

The mapping can be seen in Figure 7.1.

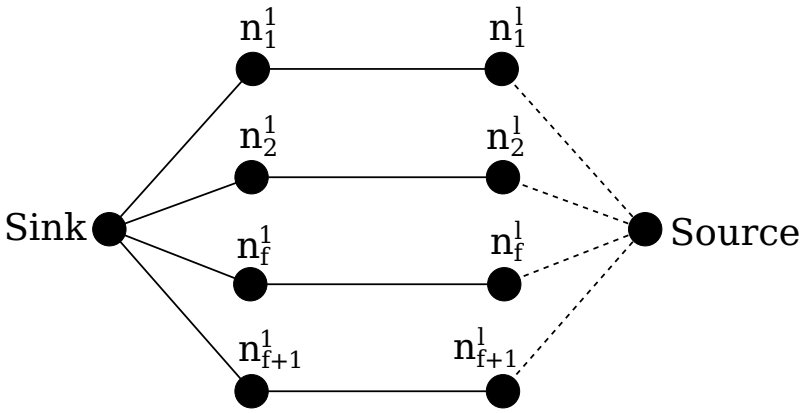


Figure 7.1: Pictorial representation of mapping.

It is now shown that a solution to DAP exists if and only if a solution to DCP exists.

$\Leftarrow$ Given a solution to DCP, attacker paths are obtained that are not capture paths by assigning slots to nodes in N as follows:

- $\mathcal{S}(s_1), \ldots, \mathcal{S}(t_1) = |V|, \ldots, |V| - length(P_1) + 1$,
- $\mathcal{S}(s_i), \ldots, \mathcal{S}(t_i), 2 \leq i \leq k = (|V| - \sum_{j=1}^{i-1} length(P_j)), \ldots, (|V| - \sum_{j=1}^{i} length(P_j) + 1)$.
- $\forall n \notin \bigcap_{i=1}^{k} P_i$, the following is done: all of the nodes in $\bigcap_{i=1}^{k} P_i$ are ranked according to the slot value under Ξ, largest first which is denoted by n_b. Then, starting with slot value $(|V| - \sum_{j=1}^{i} length(P_j))$, allocate consecutive values to all nodes in n_b's branch in a depth-first manner. A similar approach is taken for the the branch of the next highest's node (but using the set of available values) until all branches are exhausted and all nodes have been allocated a slot.

Please observe that since nodes are allocated unique slot values in a depth-first manner, both DAS and collision-freedom are satisfied.

$\Rightarrow$ This is trivial, since the existence of $f + 1$ disjoint attacker paths of length l that are not capture paths implies that, under the above mapping, the same paths are the paths from s_i to t_i.

□

At this point, one of the main results of this chapter is presented.

Theorem 4 (DAP & NP-completeness)**.** *DAP is NP-complete.*

Proof. This follows from Lemmas 7 and 8. □

In summary, for a network that is at least $(f+1)$-connected[1], the complexity of designing FT-SLP is NP-complete. FT-SLP design where the network is 2-connected is now studied. A 2-connected network is chosen as this is the minimum required to have one additional redundant path to use as a backup in case of a fault.

[1] A node is k-connected if there are k paths between the node and the sink.

7.2 Algorithms

Given the complexity of developing a fault-tolerant SLP-aware DAS in an $(f+1)$-connected network, an investigation is done into the design of a fault-tolerant SLP-aware weak DAS protocol in a 2-connected network. Unfortunately, a negative result to this problem is shown.

Theorem 5 (Impossibility of FT-SLP Weak DAS)**.** *Given (i) a network $G = (N, L)$, (ii) a children function $\mathbb{C}$, (iii) a sink-distance function δ, (iv) an SLP-aware weak DAS schedule Ξ for G, (v) a source node $\mathcal{S}$, (vi) a safety period α and (vii) f node crashes, then Ξ is not a fault-tolerant weak DAS schedule.*

Proof. This is proven by construction. Since Ξ is an SLP-aware weak DAS, then there exists no capture path in G under Ξ. Assume the following: (i) $n_2 \in \text{children}(n_1)$, (ii) $N_2 = \{n_1, n_3\}$ and (iii) $\delta(n_1) = 1$, $\delta(n_2) = 2$, $\delta(n_3) = 3$. Because Ξ is a weak DAS, then $\Xi(n_1) > \Xi(n_2) > \Xi(n_3)$. Now, assume n_1 crashes. Since Ξ is SLP-aware, then there exists no capture path in $G \setminus \{n_1\}$. Assume that Ξ is fault-tolerant weak DAS. Then, there exists another path from n_2 to V_Δ such that the next node has slot greater than n_2. However, the only path from n_2 to V_Δ is through n_3 and $\Xi(n_3) < \Xi(n_2)$. Hence, weak DAS is violated, i.e., Ξ is not fault-tolerant. □

There are two main issues behind the impossibility result: (i) when faults occur, it is difficult to satisfy both SLP and fault tolerance simultaneously and (ii) the network connectivity. For (i), the requirement for fault tolerance is weakened and require eventual fault tolerance (see Definition 9). For (ii), nodes are required to have at least two potential d-paths.

The problem of designing FT-DAS is depicted in Figure 7.2, in which Figures 7.2a and 7.2b show the creation of an a-path away from the source. However, a crash (Figure 7.2c) occurs and the a-path disappears, and in Figure 7.2d a capture path is formed instead. Hence, the impossibility of guaranteeing both fault tolerance and SLP DAS simultaneously. Subsequently, to solve this, eventual fault tolerance can only be guaranteed with SLP-aware weak DAS in a network with nodes having 2 d-paths. Figure 7.3 shows the process of a new a-path being created when a node crashes. A distributed algorithm that solves the eventual fault-tolerant SLP-aware DAS problem using this method is proposed. The protocol developed consists of five stages.

7.2.1 Phase 1: DAS Schedule

Algorithm 11 consists of the standard DAS protocol. The process begins from the sink, where each node will store a list of its potential parents, number of hops from

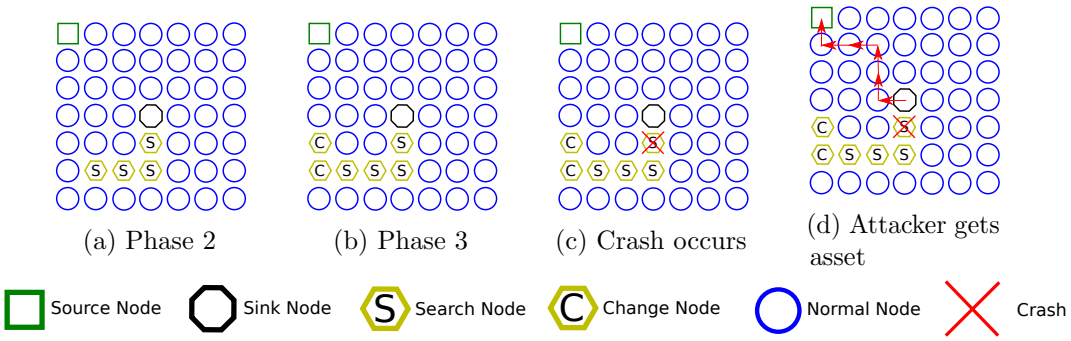


Figure 7.2: Showing the stages of the fault intolerant SLP DAS algorithm where a crash leads to the capture of the source

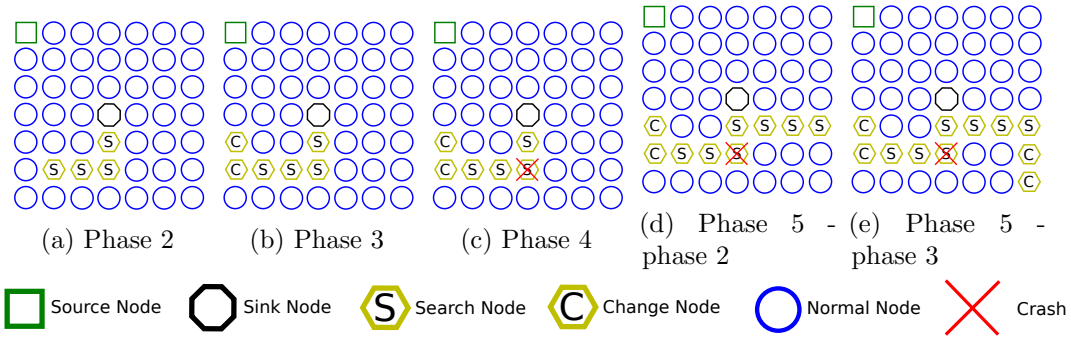


Figure 7.3: Showing the stages of the fault tolerant algorithm where a crash occurs and the diversionary path is rebuilt

Algorithm 11 Phase 1 - DAS algorithm: A slot assignment protocol for data aggregation scheduling.

```
    ▷ DAS protocol for process i
 1: variables
 2:    myN: set of int                                  ▷ Set of neighbours
 3:    Npar: set of int init ∅                          ▷ Set of potential parents
 4:    children: set of int init ∅                      ▷ Set of children
 5:    Others[]: array of set of int init ∅             ▷ Set of potential competitors
 6:    Ninfo[]: array of (int,int)                      ▷ 2-hop neighbourhood information (hop,slot)
 7:    hop, par, slot: int,int,int init ⊥, ⊥, ⊥         ▷ DAS information
 8:    Normal: bool init 1                              ▷ Flag for re-computation of DAS
 9:    start: bool init 1                               ▷ Flag to begin algorithm
10:    dissem: timer init α                             ▷ Timer for broadcasting neighbourhood information

11: constants
12:    sink: bool                                       ▷ 1 if node is sink, otherwise 0
13:    length: int                                      ▷ Length of redirection
14:    size: int                                        ▷ Size of the network

15: init :: start →                                     ▷ Sink triggers the protocol
16:    if sink then
17:       for all n ∈ myN do Ninfo[n] ← (⊥, ⊥)
18:       hop, par, slot, start ← 0, ⊥, size, 0
19:       Ninfo[i].hop, Ninfo[i].slot ← hop, slot

20: dissem :: timeout(dissem) →                         ▷ When dissem times out
21:    if slot ≠ ⊥ then
22:       broadcast DISSEM⟨Normal, {Ninfo[j] | j ∈ myN}, par⟩
23:    set⟨dissem, α⟩

    ▷ Normal dissem message received
24: receiveN :: receive DISSEM⟨1, N, p⟩ from j →
25:    if slot = ⊥ ∧ N[j].slot ≠ ⊥ then
26:       Npar ← Npar ∪ {j}
27:       Others[j] ← Others[j] ∪ {n|N[n].slot = ⊥}
28:    for all n ∈ myN | N[n] ≠ ⊥ do Ninfo[n] ← N[n]

    ▷ Non-normal (update) dissem message received
29: receiveU :: receive DISSEM⟨0, N, p⟩ from j →
30:    if par = j then
31:       if slot ≥ N[j].slot then
32:          slot ← N[j].slot
33:          Normal ← 0
34:       for all n ∈ myN | N[n] ≠ ⊥ do Ninfo[n] ← N[n]
```

Algorithm 11 (cont.) Phase 1 - DAS algorithm: A slot assignment protocol for data aggregation scheduling.

35: *process* :: **rcv**$\langle\rangle \rightarrow$ ▷ Finished receiving dissem messages
36: **if** $slot = \bot$ **then**
▷ Select parent and calculate DAS info
37: $hop \leftarrow min\{h \mid (h, s) \in Ninfo[k], k \in Npar\} + 1$
38: $par \leftarrow min\{k \mid k \in Npar, Ninfo[k] = (hop, s)\}$
39: $slot \leftarrow Ninfo[par].slot - rank(i, Others[par]) - 1$
40: $Ninfo[i] \leftarrow (hop, slot)$
41: **for all** $n \in myN \mid Ninfo[n].slot = \bot$ **do**
42: $children \leftarrow children \cup \{n\}$
▷ Detect and resolve collisions
43: **if** $\exists j, j \neq i : Ninfo[j].slot = slot$ **then**
44: **if** $hop > Ninfo[j].hop \vee (hop = Ninfo[j].hop \wedge i > j)$ **then**
45: $slot \leftarrow slot - 1$

the sink and its own time slot. Upon receiving a dissemination message, the node will select a parent from its potential parents and assign itself a slot lower than those of all nodes it has seen so far. This ensures that during phase 3 (slot refinement) the node will be able to easily select a new parent.

As each node broadcasts its neighbourhood, nodes are able to build a view of their 2-hop neighbourhood. The nodes utilise this information to detect slot collisions and resolve them.

7.2.2 Phase 2: Node Locator

The node locator phase (Algorithm 12) searches for a node of specified distance away from the sink at which to begin refining slots. The process begins at the sink and sends search messages along the path that that is known the attacker will take (sending to the node with the lowest slot in the neighbourhood). As the search messages are sent, these nodes record their position along the search path and the next node in the path (the path child).

7.2.3 Phase 3: Slot Refinement

The slot refinement phase (Algorithm 13) creates a path that redirects the attacker by altering node slot values. The phase begins at the node selected by phase 2, where change messages will be passed along *lenD* times if each node has two or more potential parents or until a node with only one potential parent is reached. The next node to continue the path is one of the current node's potential parents (excluding its actual parent). When a node alters its slot, it sets it to be the lowest slot in the neighbourhood of the previous node on the path. To ensure DAS is retained, *Normal* is set to 0 signifying an update phase for surrounding nodes. Returning to

Algorithm 12 Phase 2 - Node Locator: An algorithm that searches for a suitable location in the network for where the redirection can occur. The algorithm inherits variables from Algorithm 11.

```
    ▷ Search protocol for process i
 1: variables
 2:   startSearch: bool init 1                          ▷ Start of search
 3:   startNode: bool init 0                            ▷ First node of redirection
 4:   pr: int init ⊥                                    ▷ Length of redirection
 5:   from: set of int init ∅                           ▷ To prevent backtracking
 6:   pathChild: int init ⊥                             ▷ Next node in redirection path
 7:   pathOrder: int init ⊥                             ▷ Order node is in full path

 8: constants
 9:   dist: int init SD                                 ▷ Length of search path
10:   diam: int init δ                                  ▷ Diameter of network

11: startS :: startSearch ∧ sink →                      ▷ Begin search process
12:   for all c ∈ children do
13:     if Ninfo[c].slot = min{Ninfo[k].slot | k ∈ children} then
14:       startSearch ← 0
15:       pathChild ← c
16:       pathOrder ← 0
17:   broadcast SEARCH⟨pathChild, dist, pathOrder⟩

    ▷ Receive search message
18: receiveS :: receive SEARCH⟨k, d, po⟩ from j →
19:   from ← from ∪ {j}
20:   if i = k then
21:     pathOrder ← po + 1
22:     if (d = 0) ∧ (Npar \ {par, j} ≠ ∅) then
23:       startNode ← 1
24:       pr ← ⌈dist/3⌉
25:     if (d = 0) ∧ (Npar \ {par, j} = ∅) then
26:       if children ≠ ∅ then
27:         pathChild ← choose(children)
28:       else
29:         pathChild ← choose(myN \ {par})
30:       broadcast SEARCH⟨pathChild, d, pathOrder⟩
31:     if d > 0 then
32:       for all c ∈ children do
33:         if Ninfo[c].slot = min{Ninfo[k].slot | n ∈ children} then
34:           pathChild ← c
35:       broadcast SEARCH⟨pathChild, d − 1, pathOrder⟩
```

Algorithm 11, when a node receives a dissemination with the update flag set, it will check that its slot value is lower than its parent, lowering the slot value if necessary.

7.2.4 Phase 4: Failure Detection and Notification

Algorithm 14 shows the failure detection and notification phase. Utilising the same *dissem* timer as phase 1, the algorithm begins by confirming that any nodes left in the *suspects* set has crashed and removes all of them from the memory of the node. If any of the suspects are the parent of the node, the node will attempt to change to another of its potential parents. Failing that results in the node resetting and waiting to be reassigned information from the process described in Phase 1. After this, the node broadcasts a crash message containing its ID. If the path child has crashed, a flag is set to signal a reconstruction is required. Finally, the suspects list is refreshed to contain all nodes in the one-hop neighbourhood.

When a dissemination message (from phase 1) is received, the node that sent it is removed from the suspects list. As such, if a dissemination message is not received from a node in a period, it is assumed that the node has crashed.

Upon receiving a crash message, if that node is the parent of the receiving node then, as before, the node will attempt to change parent to a different potential parent, failing that resetting. Another crash message is then sent to alert the children of the current node that the slot has been altered.

7.2.5 Phase 5: SLP Path Reconstruction

The path reconstruction (Algorithm 15) occurs when phase 4 flags that it is required. Using the stored position in the path, the parent of the crashed node will calculate how many messages need to be sent and whether it is search or change messages that are required. The correct message with parameters set is then sent again, rebuilding the path using the methods in phases 2 and 3.

Note that the algorithm will not guarantee a maximal privacy level as the initial slot assignment is done with no knowledge of the source location. It is possible that the diversion leads an attacker closer to the source. To prevent this, information about application messages routing is required, which is typically not available at the MAC level being targeted. To obtain routing level information, cross layer techniques are required.

Algorithm 13 Phase 3 - Slot Refinement: An algorithm that refines the original slot assignment. The algorithm inherits variables from Algorithms 11 and 12.

▷ Slot refinement protocol for process i

1: $startR :: startNode \rightarrow$ ▷ Begin the change process
2: $startNode \leftarrow 0$
3: $pathChild \leftarrow choose(Npar \setminus \{par\} \setminus from)$
4: $nSlot \leftarrow \min(\{Ninfo[j].slot \mid j \in myN\} \cup \{slot\})$
5: **broadcast** CHANGE$\langle pathChild, nSlot, pr - 1, pathOrder\rangle$

6: $receiveC ::$ **receive** CHANGE$\langle p, s, d, po\rangle$ **from** $j \rightarrow$
7: **if** $d > 0 \wedge i = p$ **then**
8: **if** $myN \setminus \{par\} \setminus from \neq \emptyset$ **then**
9: $pathChild \leftarrow choose(myN \setminus \{par\} \setminus from)$
10: $slot \leftarrow s - 1$
11: $Ninfo[i] \leftarrow (hop, slot)$
12: $nSlot \leftarrow \min(\{Ninfo[k].slot \mid k \in myN\} \cup \{slot\})$
13: **broadcast** CHANGE$\langle pathChild, nSlot, d - 1, pathOrder\rangle$
14: **if** $d = 0 \wedge i = p \wedge (myN \setminus \{par\} \setminus from \neq \emptyset)$ **then**
15: $Normal, slot \leftarrow 0, s - 1$
16: $Ninfo[i] \leftarrow (hop, slot)$

Algorithm 14 Phase 4 - Failure Detection and Notification: An algorithm that detects crashed nodes/broken links and repairs DAS. The algorithm inherits variables from Algorithms 11, 12 and 13.

```
    ▷ Strongly complete failure detector for process i
 1: variables
 2:   suspects: set of int init myN                          ▷ Crash suspects
 3:   pathReconstruct: bool init 0                  ▷ Should reconstruct redirection path

    ▷ Crash news received from parent, so reset
 4: crashN :: receive CRASH⟨⟩ from j →
 5:   if par = j then
 6:     if Npar ≠ ∅ then
 7:       par ← ∃p ∈ Npar | p ≠ par
 8:       hop ← Ninfo[par].hop + 1
 9:       slot ← Ninfo[par].slot − rank(i, Others[par]) − 1
10:     else
11:       par, slot, hop := ⊥, ⊥, ⊥
12:     Ninfo[i] := (hop, slot)
13:     broadcast CRASH⟨⟩

    ▷ Determine if node alive by whether they send a dissem message
14: filter :: receive DISSEM⟨n, N, p⟩ from j →
15:   suspects ← suspects \ {j}

16: crashD :: timeout(dissem) →                          ▷ Check for crashes
17:   for all s ∈ suspects do
18:     myN ← myN \ {s}
19:     Npar ← Npar \ {s}
20:     children ← children \ {s}
21:     Ninfo ← Ninfo \ {s}
22:     Others ← Others \ {s}
23:     for all o ∈ Others do Others[o] ← Others[o] \ {s}
24:   if par ∈ suspects then
25:     if Npar ≠ ∅ then
26:       par ← ∃p ∈ Npar | p ≠ par
27:       hop ← Ninfo[par].hop + 1
28:       slot ← Ninfo[par].slot − rank(i, Others[par]) − 1
29:     else
30:       par, slot, hop := ⊥, ⊥, ⊥
31:     Ninfo[i] ← (hop, slot)
32:     broadcast CRASH⟨⟩
33:   if pathChild ∈ suspects then
34:     pathReconstruct ← 1
35:   suspects ← myN                          ▷ Reset suspects after processing
```

Algorithm 15 Phase 5 - Path Reconstruction: An algorithm that, upon detection of a failure from a node in the redirection path, rebuilds the path from the parent of that node. The algorithm inherits variables from Algorithms 11, 12, 13, and 14.

▷ Begin reconstruction protocol for process i
1: $startP :: pathReconstruct \rightarrow$
2: $pathReconstruct \leftarrow 0$
3: **if** $pathOrder < dist$ **then** ▷ If node on search path
4: **for all** $c \in children$ **do**
5: **if** $Ninfo[c].slot = \min\{Ninfo[k].slot \mid k \in children\}$ **then**
6: $startSearch, pathChild := 0, c$
7: $d \leftarrow dist - pathOrder$
8: **broadcast** SEARCH$\langle pathChild, d, pathOrder\rangle$
9: **else** ▷ Otherwise node is on change path
10: $pathChild \leftarrow choose(Npar \setminus \{par\} \setminus from)$
11: $nSlot \leftarrow \min(\{Ninfo[j].slot \mid j \in myN\} \cup \{slot\})$
12: $d \leftarrow dist + \lceil dist/3 \rceil - pathOrder$
13: **broadcast** CHANGE$\langle pathChild, nSlot, d, pathOrder\rangle$

7.3 Experimental Setup

7.3.1 Network Configuration

The network configuration is as described in Section 4.2.

It should be noted that although this solution requires a 2-connected network, some aspects of the configuration do not comply with this requirement, namely the nodes straight vertically and horizontally from the sink node are only 1-connected when operating under the ideal communication model. The consequence of this is that should a crash occur in one of these nodes, then it is not possible to have a backup route around the node. While this can be problematic and reduce the efficacy of the solution on this configuration, the remainder of the network is acceptable. This issue has not drastically affected the results as can be seen later in the chapter. The significance of this issue is reduced if used in a real-world deployment (and even in the low-asymmetry communication model) where the degree of connectivity is increased but must still be a consideration for unusually sparse network topologies.

7.3.2 Simulation Parameters

In order to compare SLP TDMA DAS and FT SLP TDMA DAS, the SLP TDMA DAS results from Chapter 6 are used and presented next to FT SLP TDMA DAS that operates in a simulation with node crashes. The node crashes shall occur with a 50% probability along the initial diversionary route. The reason for restricting these crashes to exclusively occur along the diversionary route is that this is the path that will have to be rebuilt in order to divert the attacker, any other crashes not on this route will have no effect on the final result. With crashes only on the diversionary route, a comparison can be made to see if the fault-tolerant algorithm provides similar metrics to ordinary SLP TDMA DAS. Should the metrics be similar, they show that fault-tolerant guarantees can be provided without much loss of efficiency.

The majority of the aspects of the simulation are the same as described in Section 4.3, with the only difference being that the low-asymmetry communication model is not used. This is because the efficacy of the algorithm should be shown in the presence of *node crashes only*. This will show a clear comparison between the two algorithms, with node crashes as the only changing variable.

7.3.3 Algorithm Parameters

The parameters for the algorithm are the same as those defined in Table 6.1, Subsection 6.3.1, with the only difference being that the number of slots available has been increased to 200. This also increases the source period to 25 seconds. The reason for

this alteration is to ensure that the algorithm has enough slots to operate correctly for even the largest network size. The fault tolerant algorithm is expected to use additional slots due to the method of updating slot values on node crashes and link failures, and providing this many slots ensures correct operation.

7.3.4 Attacker Model

The attacker model is the same as described in the previous chapter (Subsection 6.3.2). That is, the attacker model is implemented as described in Section 3.2, with an implementation satisfying the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker. Again, it is assumed that the attacker is aware if the period length and thus only performs one move per period.

No other attacker model parametrisations are to be tested, due to the rationale provided in Subsection 6.4.1, which states that the $(1, 0, 1, s_0, D)$-$\mathcal{A}$ attacker is the strongest or representative of the strongest attacker available using this model.

7.4 Results

7.4.1 Expected Outcomes

The expectation is that the majority of metrics will remain similar to the baseline SLP TDMA DAS, perhaps performing slightly worse due to the crashes and the time taken to fix the path. However, the overhead of change, search and crash messages will increase in FT SLP TDMA DAS due to having to create the diversionary route more than once.

7.4.2 Comparison of FT SLP TDMA DAS and SLP TDMA DAS

The plots in Figure 7.4 show the comparisons for each metric when the algorithms are using different search distances.

Figure 7.4a shows a comparison of the capture ratios between the algorithms. Averagely, FT SLP TDMA DAS has a marginally higher capture ratio than ordinary SLP TDMA DAS. This is due to the process of rebuilding the diversionary route and potentially having the new route lead closer to the source (as the algorithm has no knowledge of source location).

Figure 7.4b shows that FT SLP TDMA DAS has a higher message send rate than SLP TDMA DAS. The reason for this is more accurately displayed in Figure 7.4e, which shows the additional overhead of all SLP-related messages (including crash messages). As the diversionary route may have to be built multiple times, and FT

SLP TDMA DAS has a system to report crashes along the route, this leads to more messages being sent. However, this increase was predicted in Subsection 7.4.1.

Figure 7.4c shows that between both algorithms, the received ratio (or delivery ratio) is unchanged. This result is positive as no normal messages are lost in the process of causing the crashes and rebuilding the diversionary route or paths to the sink.

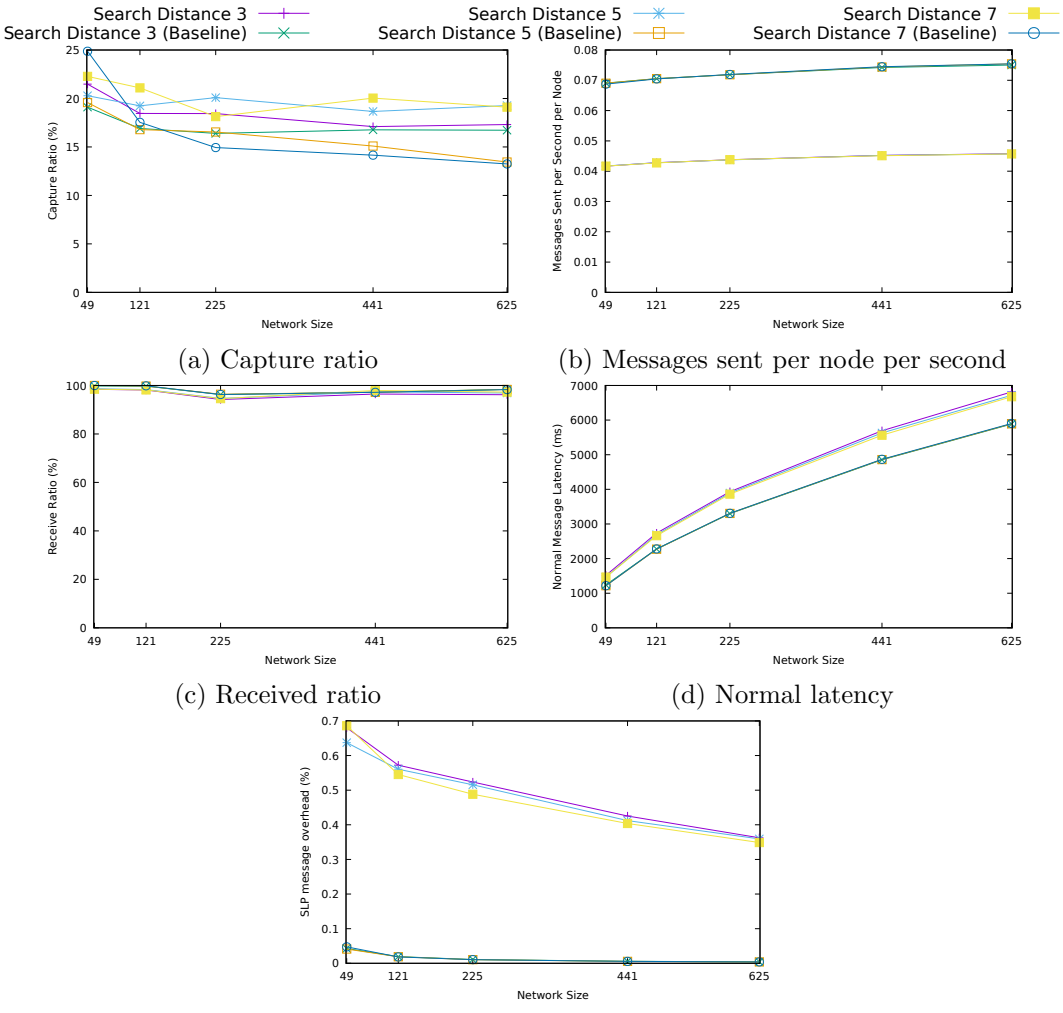
Figure 7.4d is also unsurprising as it was predicted that FT SLP TDMA DAS utilises more slots than SLP TDMA DAS. The plot shows that this increase in slot utilisation is not a dramatic increase, with 1000ms on the plot equating to ten additional slots (due to the slot period of 0.1s). As described in Subsection 6.4.1, the normal message latency is still bounded by the number of slots multiplied by the slot period.

7.4.3 Comparison with other SLP solutions

Similarly to the previous chapter, it is not realistic to make comparisons between MAC layer solutions and routing layer solutions (such as ILPRouting and DynamicSPR) when the latter has access to significantly more information about the network than the former. Additionally, no other SLP protocol provides fault tolerance in current works which is a significant factor when making comparisons.

7.5 Conclusion

This chapter has provided a number of contributions. The fault-tolerant DAS problem is shown to be NP-complete. A version of fault-tolerant SLP DAS is created called eventually fault-tolerant SLP-aware DAS and five phases of the distributed algorithm are provided alongside this. This algorithm has been implemented and tested in simulation to show that FT SLP TDMA DAS has only a marginal decrease in performance compared to SLP TDMA DAS as well as providing the fault-tolerant guarantees.



(a) Capture ratio

(b) Messages sent per node per second

(c) Received ratio

(d) Normal latency

(e) SLP overhead

Figure 7.4: Comparison of FT against standard SLP TDMA DAS (Baseline)

Chapter 8

Conclusions and Future Work

This research has presented methods to provide Source Location Privacy through the adjustment of TDMA schedules at the MAC level. The aim of this chapter is to discuss some of the common patterns and considerations that have arisen through performing this research, present some options as to how the presented methods and techniques can be expanded in future work and to summarise the work presented in this thesis.

8.1 Discussion

A number of common patterns have emerged throughout this thesis. This section discusses those patterns.

8.1.1 Network Size

As can be seen throughout the results presented in Chapters 6 and 7, an increase in network size offers an increase in SLP. This is because, with a larger network, there are more opportunities to create diversionary routes than in smaller networks which makes it less likely that any path will lead to the source. In both chapters, the size of the diversionary route was equal to the sink-source distance. With a larger sink-source distance, the diversionary path is longer and has the chance to lead the attacker much farther away from the source.

8.1.2 Routing Protocol

Throughout the thesis, the routing protocol used for normal messages is simple flooding. This is because it provides no SLP whatsoever and as such is good to build a MAC protocol under as all SLP provided is known to come from the MAC layer changes. The reason for flooding providing no SLP is that it leaks information to an

attacker no matter where they are in the network. However, flooding is not energy efficient as it sends a very large number of messages and it can drop messages due to a lack of retransmissions. As such, it is not suitable for any real-world deployment of these methods. Using another routing protocol on top of the MAC layer is a strong possibility.

Realistically, due to the way the MAC layers have been defined, a combined routing-MAC cross-layer solution may provide a higher efficiency. This of course is more complex than just altering the routing protocol, but would most likely provide better results.

8.1.3 Considering Other Metrics in Design

The reasoning behind the designs in this thesis is to provide SLP for low message overhead. This section considers prioritising different criteria at the design stages. This is not to say that other metrics were not considered at all, but to provide an alternative view of if a different metric was the highest priority.

Latency is often considered an important aspect of a WSN deployment. Being able to get data as quickly as possible is a highly desirable property, such as in military applications. While DAS is also designed to reduce latency over other TDMA solutions, it is still slower than the majority of Carrier-Sense Multiple Access (CSMA) designs because these designs don't have to wait for a period to get the message through the network. If latency was the biggest concern, TDMA is probably not the best choice. However, given completely optimal slot utilisation, DAS would be the lowest latency TDMA implementation.

Another desirable quality is network throughput. Some applications can be extremely data-intensive, making a higher throughput in the network a must. The throughput of algorithms presented in this thesis has not been discussed. However, DAS is designed to have parent nodes aggregate data from its children before sending it on towards the sink. The extent to which the data can be aggregated is of great importance when considering throughput. Disregarding aggregation, the network throughput of these algorithms is not very high, as only a single message can be sent in a slot and then the node must wait for an entire period before it can send again. This can be perfectly acceptable because if the asset monitoring network is only tracking the location of the asset (and possibly some other information), then a single message of aggregated data per TDMA period is often sufficient.

There will always be some form of trade-off between the metrics in every design, so selecting the correct one for the application is critical.

8.1.4 Localised TDMA

The TDMA used in this thesis has assigned slots in a global fashion across the entire network. Localising the TDMA implementation by splitting the network into local groupings and performing a separate slot assignment for each group could have significant benefits. For instance, the number of slots required would be lowered leading to shorter TDMA periods and lower latencies. It would also significantly increase the scalability of the algorithms. However, this would add significant complexity as each local sink in a group would have to communicate with the real sink in the network, i.e. forwarding data onwards. Routing algorithms for networks with clusters are not uncommon, so an existing implementation could be taken and modified to fit the purpose.

8.2 Future Work

8.2.1 Impact of Network Configuration

While this thesis examined SLP at different network sizes, other aspects of the configuration were not explored.

The topology of the network used was static throughout this research. That is, this thesis presented varying sizes of grid networks with the sink in the centre and the source in one corner. Expansion of this work could examine the produced algorithms and methods on different network layouts.

Multiple sinks could be also be introduced and tested with the existing algorithms. This would show that the algorithms could cope with large-scale, multi-sink networks.

8.2.2 Handling Different Attackers

While the model used for an attacker in this thesis is fairly expressive, realistically it does not cover a fraction of what an attacker could be capable of in a real-world deployment.

Multiple Attackers

Multiple attackers could be used simultaneously in a simulation. This could be as simple as employing two attackers with differing parametrisations at the same time or a significantly more sophisticated model where the attackers could share information, more akin to a real-world scenario.

Beyond Eavesdropping

Until now, it has been assumed that attackers only have the capability to eavesdrop on node communications. Of course, this is not always the case, as in other cases attackers can destroy nodes or alter their behaviour, interfere with network traffic (such as blocking or replaying messages) and generally be far more intrusive in network operation. This expands into a huge area of research, where many options on future work could be taken.

Attacker Starting Location

In the current setup, the attacker begins at the sink node. While this is a logical place for an attacker to begin, as a message from the source must pass through that location, an attacker may choose a different location to begin. The current state of the algorithms creates a diversionary route from the sink node outwards into the network and should an attacker choose a different location this defence would be moot. Changing the starting location would require different algorithms in order to combat such behaviour.

8.2.3 Testbed Deployment

As described in Section 4.5, it was not possible to select a testbed that could operate any of the algorithms properly, as they were simply too small and too dense. However, this does not preclude that a suitable testbed would not become available in the future. While simulations using the low-asymmetry communication model were performed (which represents a more realistic communication environment), a testbed deployment would provide more evidence that these algorithms do operate correctly in the real-world.

8.2.4 Cross-Layer Solutions

While this thesis focused purely on MAC layer SLP protection, this could be significantly improved by incorporating routing layer information into a solution. Using MAC and routing together provides a more detailed view of the network while simultaneously allowing very low-level methods to be deployed.

8.3 Conclusion

This thesis has focused on developing a MAC level solution to the SLP problem by manipulating TDMA schedules to divert an attacker. Using an offline method, it is possible to provide a near 0% capture ratio while using online methods it is possible

to reduce the capture ratio from a protectionless TDMA DAS. The distributed online method was also improved to provide fault-tolerant guarantees for very little cost.

Initially, using evolutionary computation, schedules can be built offline to provide a near-perfect level of SLP. This genetic algorithm produces the schedules over generations of potential solutions to find ones that both minimise slot utilisation (thus decreasing latency) and also maximising the level of SLP provided.

As a response to the offline method, an online, distributed algorithm was created in order to provide some MAC level SLP during network operation. This does not provide the level of SLP that the first method does, but is an improvement over a protectionless version for almost no message overhead.

Finally, as WSNs tend to be unreliable, a fault-tolerant version of the previous algorithm was produced, providing fault-tolerant guarantees for only a minor performance cost.

In summary, this thesis shows that it is possible to create full SLP solutions using only the MAC layer in the network stack. Using TDMA schedule manipulation to re-engineer traffic patterns, three different methods were created, one offline and two online. A number of metrics were examined showing the performance of these methods. This opens the concept of future work to improved performance using cross-layer SLP solutions that employ multiple methods of protection.

Appendix A

Result Reproduction

The results presented in this thesis were obtained in such a fashion that they could easily be reproduced. The results of the simulations are deterministic for a given random seed, allowing runs of the simulation utilising the same seed to produce identical results.

As part of this research, an existing framework to perform SLP simulations using TOSSIM was used and can be found at `github.com/MBradbury/slp`. This framework was updated collaboratively, with specific additions being made (including the source code for algorithms created for this thesis) which has been merged into the same repository. Instructions on operating the framework can also be found there.

The source code for the genetic algorithm presented in Chapter 5 can be found at `github.com/jack-kirton/slp-tdma-das-genetic`.

Bibliography

[1] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru, and A. Wood. Envirotrack: towards an environmental computing paradigm for distributed sensor networks. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 582–589, 2004. doi: 10.1109/ICDCS.2004.1281625.

[2] Kemal Akkaya and Mohamed Younis. An energy-aware QoS routing protocol for wireless sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, ICDCSW '03, pages 710–715, Washington, DC, USA, May 2003. IEEE Computer Society. ISBN 0-7695-1921-0. doi: 10.1109/ICDCSW.2003.1203636.

[3] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005. ISSN 1570-8705. doi: 10.1016/j.adhoc.2003.09.010. URL `http://www.sciencedirect.com/science/article/pii/S1570870503000738`.

[4] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, December 2004. ISSN 1536–1284. doi: 10.1109/MWC.2004.1368893.

[5] Aboli Arun Anasane and Rachana Anil Satao. A survey on various multipath routing protocols in wireless sensor networks. *Procedia Computer Science*, 79(Supplement C):610 – 615, 2016. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2016.03.077. Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016.

[6] B. H. Arabi. Solving np-complete problems using genetic algorithms. In *2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation (UKSim)*, pages 43–48, April 2016. doi: 10.1109/UKSim.2016.65.

[7] Th. Arampatzis, J. Lygeros, and S. Manesis. A survey of applications of wireless sensors and wireless sensor networks. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 719–724, June 2005. doi: 10.1109/.2005.1467103.

[8] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Comput. Netw.*, 46(5):605–634, December 2004. ISSN 1389-1286. doi: 10.1016/j.comnet.2004.06.007.

[9] Mahesh Arumugam. A distributed and deterministic TDMA algorithm for write-all-with-collision model. In *Proceedings of the 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, SSS '08, pages 4–18, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-89334-9. doi: 10.1007/978-3-540-89335-6_4.

[10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010. ISSN 1389-1286. doi: 10.1016/j.comnet.2010.05.010.

[11] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004. ISSN 1545-5971. doi: 10.1109/TDSC.2004.2.

[12] Ioannis Avramopoulos, Hisashi Kobayashi, Randolph Wang, and Arvind Krishnamurthy. Highly secure and efficient routing. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, pages –208, 2004. doi: 10.1109/INFCOM.2004.1354494.

[13] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. Radio link quality estimation in wireless sensor networks: A survey. *ACM Trans. Sen. Netw.*, 8(4):34:1–34:33, September 2012. ISSN 1550-4859. doi: 10.1145/2240116.2240123.

[14] B. Bates, A. Keating, and R. Kinicki. Energy analysis of four wireless sensor network MAC protocols. In *Wireless and Pervasive Computing (ISWPC), 2011 6th International Symposium on*, pages 1–6, February 2011. doi: 10.1109/ISWPC.2011.5751321.

[15] Zinaida Benenson, Peter M. Cholewinski, and Felix C. Freiling. *Wireless Sensors Networks Security*, chapter Vulnerabilities and Attacks in Wireless Sensor Networks, pages 22–43. IOS Press, 2008.

[16] L. Benini, D. Brunelli, C. Petrioli, and S. Silvestri. Genesi: Green sensor networks for structural monitoring. In *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*, pages 1–3, June 2010. doi: 10.1109/SECON.2010.5508230.

[17] Ketki Ram Bhakare, RK Krishna, and Samiksha Bhakare. An energy-efficient grid based clustering topology for a wireless sensor network. *International Journal of Computer Applications*, 39(14):24–28, 2012.

[18] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACaw: A media access protocol for wireless LAN's. *SIGCOMM Comput. Commun. Rev.*, 24(4):212–225, October 1994. ISSN 0146-4833. doi: 10.1145/190809.190334.

[19] Edoardo S. Biagioni and K. W. Bridges. The application of remote sensor technology to assist the recovery of rare and endangered species. *International Journal of High Performance Computing Applications*, 16:2002, 2002.

[20] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, DIALM '99, pages 48–55, New York, NY, USA, 1999. ACM. ISBN 1-58113-174-7. doi: 10.1145/313239.313282.

[21] Matthew Bradbury and Arshad Jhumka. A near-optimal source location privacy scheme for wireless sensor networks. In *16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 409–416, August 2017. doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.265.

[22] Matthew Bradbury and Arshad Jhumka. Understanding source location privacy protocols in sensor networks via perturbation of time series. In *INFOCOM, 2017 Proceedings IEEE*, pages 1611–1619, May 2017. doi: 10.1109/INFOCOM.2017.8057122.

[23] Matthew Bradbury, Matthew Leeke, and Arshad Jhumka. A dynamic fake source algorithm for source location privacy in wireless sensor networks. In *14th IEEE International Conference on Trust, Security and Privacy in Computing*

and Communications (TrustCom), pages 531–538, August 2015. doi: 10.1109/Trustcom.2015.416.

[24] Matthew Bradbury, Arshad Jhumka, and Matthew Leeke. Hybrid online protocols for source location privacy in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 115:67–81, May 2018. ISSN 0743-7315. doi: 10.1016/j.jpdc.2018.01.006.

[25] Matthew Bradbury, Arshad Jhumka, and Carsten Maple. The impact of decreasing transmit power levels on flocklab to achieve a sparse network. In *Proceedings of the 2Nd Workshop on Benchmarking Cyber-Physical Systems and Internet of Things*, CPS-IoTBench '19, pages 7–12, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6693-9. doi: 10.1145/3312480.3313171. URL `http://doi.acm.org/10.1145/3312480.3313171`.

[26] M. Brownfield, Yatharth Gupta, and N. Davis. Wireless sensor network denial of sleep attack. In *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 356–364, June 2005. doi: 10.1109/IAW.2005.1495974.

[27] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4^{th} international conference on Embedded networked sensor systems*, SenSys '06, pages 307–320, New York, NY, USA, 2006. ACM. ISBN 1-59593-343-3. doi: 10.1145/1182807.1182838.

[28] M. Ceriotti, L. Mottola, G.P. Picco, A.L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *Information Processing in Sensor Networks, 2009. IPSN 2009. International Conference on*, pages 277–288, April 2009.

[29] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Comput. Commun. Rev.*, 31(2 supplement): 20–41, April 2001. ISSN 0146-4833. doi: 10.1145/844193.844196.

[30] M.J. Chae, H.S. Yoo, J.Y. Kim, and M.Y. Cho. Development of a wireless sensor network system for suspension bridge health monitoring. *Automation in Construction*, 21(0):237–252, 2012. ISSN 0926-5805. doi: 10.1016/j.autcon.2011.06.008.

[31] G. Chakraborty and Y. Hirano. Genetic algorithm for broadcast scheduling in packet radio networks. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 183–188, May 1998. doi: 10.1109/ICEC.1998.699498.

[32] Jae-Hwan Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(4):609–619, August 2004. ISSN 1063-6692. doi: 10.1109/TNET.2004.833122.

[33] X. Chen, K. Makki, K. Yen, and N. Pissinou. Sensor network security: a survey. *IEEE Communications Surveys Tutorials*, 11(2):52–73, Second 2009. ISSN 1553-877X. doi: 10.1109/SURV.2009.090205.

[34] G. Chiandussi, M. Codegone, S. Ferrero, and F.E. Varesio. Comparison of multi-objective optimization methodologies for engineering applications. *Computers & Mathematics with Applications*, 63(5):912 – 942, 2012. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2011.11.057. URL `http://www.sciencedirect.com/science/article/pii/S0898122111010406`.

[35] Pietro Ciciriello, Luca Mottola, and GianPietro Picco. Efficient routing from multiple sources to multiple sinks in wireless sensor networks. In Koen Langendoen and Thiemo Voigt, editors, *Wireless Sensor Networks*, volume 4373 of *Lecture Notes in Computer Science*, pages 34–50. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-69829-6. doi: 10.1007/978-3-540-69830-2_3.

[36] M. Conti, J. Willemsen, and B. Crispo. Providing source location privacy in wireless sensor networks: A survey. *IEEE Communications Surveys and Tutorials*, 15(3):1238–1280, 2013. ISSN 1553-877X. doi: 10.1109/SURV.2013.011413.00118.

[37] H. Darji and H. B. Shah. Genetic algorithm for energy harvesting-wireless sensor networks. In *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 1398–1402, May 2016. doi: 10.1109/RTEICT.2016.7808061.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002. ISSN 1089-778X. doi: 10.1109/4235.996017.

[39] Jing Deng, Richard Han, and Shivakant Mishra. Secure code distribution in dynamically programmable wireless sensor networks. In *Proceedings of the 5th*

international conference on Information processing in sensor networks, IPSN '06, pages 292–300, New York, NY, USA, 2006. ACM. ISBN 1-59593-334-4. doi: 10.1145/1127777.1127822.

[40] M. Dhanaraj and C. Siva Ram Murthy. On achieving maximum network lifetime through optimal placement of cluster-heads in wireless sensor networks. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 3142–3147, 2007. doi: 10.1109/ICC.2007.521.

[41] E. W. Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11), 1974.

[42] Shlomi Dolev. *Self-stabilization*. MIT Press, Cambridge, MA, USA, 2000. ISBN 0-262-04178-2.

[43] Mianxiong Dong, Kaoru Ota, and Anfeng Liu. Preserving source-location privacy through redundant fog loop for wireless sensor networks. In *13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pages 1835–1842, Liverpool, UK, October 2015. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.274.

[44] S. M. Duan, J. L. Mao, F. H. Xiang, and H. P. Liu. Quantum genetic algorithm optimization in tdma time slot allocation for wsn. In *Control Conference (CCC), 2013 32nd Chinese*, pages 7377–7382, July 2013.

[45] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, 2004. doi: 10.1109/LCN.2004.38.

[46] Adam Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, MobiSys '03, pages 85–98, New York, NY, USA, 2003. ACM. doi: 10.1145/1066116.1066118.

[47] Adam Dunkels. A low-overhead script language for tiny networked embedded systems. Technical Report T2006:15, Swedish Institute of Computer Science, September 2006. URL `http://www.sics.se/~adam/dunkels06lowoverhead.pdf`.

[48] Adam Dunkels, Juan Alonso, and Thiemo Voigt. Making TCP/IP Viable for Wireless Sensor Networks. *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN2004)*, 2004.

[49] Adam Dunkels, Niclas Finne, Joakim Eriksson, and Thiemo Voigt. Run-time dynamic linking for reprogramming wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 15–28, New York, NY, USA, 2006. ACM. ISBN 1-59593-343-3. doi: 10.1145/1182807.1182810.

[50] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 29–42, New York, NY, USA, 2006. ACM. ISBN 1-59593-343-3. doi: 10.1145/1182807.1182811.

[51] Adam Dunkels, Fredrik Österlind, and Zhitao He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, SenSys '07, pages 335–349, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-763-6. doi: 10.1145/1322263.1322295.

[52] Adam Dunkels, Fredrik Österlind, Nicolas Tsiftes, and Zhitao He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors*, EmNets '07, pages 28–32, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-694-3. doi: 10.1145/1278972.1278979.

[53] Mathilde Durvy, Julien Abeillé, Patrick Wetterwald, Colin O'Flynn, Blake Leverett, Eric Gnoske, Michael Vidales, Geoff Mulligan, Nicolas Tsiftes, Niclas Finne, and Adam Dunkels. Making sensor networks IPv6 ready. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 421–422, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-990-6. doi: 10.1145/1460412.1460483.

[54] Vladimir Dyo, Stephen A. Ellwood, David W. Macdonald, Andrew Markham, Niki Trigoni, Ricklef Wohlers, Cecilia Mascolo, Bence Pásztor, Salvatore Scellato, and Kharsim Yousef. Wildsensing: Design and deployment of a sustainable sensor network for wildlife monitoring. *ACM Trans. Sen. Netw.*, 8(4):29:1–29:33, September 2012. ISSN 1550-4859. doi: 10.1145/2240116.2240118.

[55] Cheng Tien Ee, Rodrigo Fonseca, Sukun Kim, Daekyeong Moon, Arsalan Tavakoli, David Culler, Scott Shenker, and Ion Stoica. A modular network layer for sensorsets. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, pages 249–262, Berkeley, CA, USA, 2006. USENIX Association. ISBN 1-931971-47-1.

[56] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 263–270, New York, NY, USA, 1999. ACM. ISBN 1-58113-142-9. doi: 10.1145/313451.313556.

[57] D. Evans and D. Larochelle. Improving security using extensible lightweight static analysis. *Software, IEEE*, 19(1):42–51, 2002. ISSN 0740-7459. doi: 10.1109/52.976940.

[58] David Fotue, Guy Tanonkou, and Thomas Engel. An ad-hoc wireless sensor networks with application to air pollution detection. In Gordon K. Lee, editor, *Proceedings of the ISCA First International Conference on Sensor Networks and Applications (SNA-2009), November 4-6, 2009, Hilton San Francisco Fisherman s Wharf, San Francisco, California, USA*, pages 48–53. ISCA, 2009.

[59] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, PLDI '03, pages 1–11, New York, NY, USA, 2003. ACM. ISBN 1-58113-662-5. doi: 10.1145/781131.781133.

[60] Hock Guan Goh, Moh Lim Sim, and Hong Tat Ewe. Energy efficient routing for wireless sensor networks with grid topology. In *International Conference on Embedded and Ubiquitous Computing*, pages 834–843. Springer, 2006.

[61] A.J. Goldsmith and S.B. Wicker. Design challenges for energy-constrained ad hoc wireless networks. *Wireless Communications, IEEE*, 9(4):8–27, 2002. ISSN 1536-1284. doi: 10.1109/MWC.2002.1028874.

[62] Chen Gu, Matthew Bradbury, Arshad Jhumka, and Matthew Leeke. Assessing the performance of phantom routing on source location privacy in wireless sensor networks. In *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 99–108, November 2015. doi: 10.1109/PRDC.2015.9.

[63] Chen Gu, Matthew Bradbury, Jack Kirton, and Arshad Jhumka. A decision theoretic framework for selecting source location privacy aware routing protocols in wireless sensor networks. *Future Generation Computer Systems*, 87:514 – 526, 2018. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2018.01.046. URL `http://www.sciencedirect.com/science/article/pii/S0167739X17317028`.

[64] V.C. Gungor and G.P. Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE Transactions on*, 56(10):4258–4265, 2009. ISSN 0278-0046. doi: 10.1109/TIE.2009.2015754.

[65] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[66] Yingchao Han, Hongmei Li, and Jinghui Qiu. The analysis and summary about energy saving technologies of wireless sensor network. In *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, volume 2, pages 883–885, August 2011. doi: 10.1109/EMEIT.2011.6023235.

[67] Carl Hartung, Richard Han, Carl Seielstad, and Saxon Holbrook. Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *Proceedings of the 4^{th} international conference on Mobile systems, applications and services*, MobiSys '06, pages 28–41, New York, NY, USA, 2006. ACM. ISBN 1-59593-195-3. doi: 10.1145/1134680.1134685.

[68] A. Hassanzadeh, R. Stoleru, and Jianer Chen. Efficient flooding in wireless sensor networks secured with neighborhood keys. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7^{th} International Conference on*, pages 119–126, October 2011. doi: 10.1109/WiMOB.2011.6085415.

[69] M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, 2007. MASS 2007.*, pages 1–6, October 2007. doi: 10.1109/MOBHOC.2007.4428702.

[70] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, October 2002. ISSN 1536-1276. doi: 10.1109/TWC.2002.804190.

[71] D. Herbert, Yung-Hsiang Lu, S. Bagchi, and Zhiyuan Li. Detection and repair of software errors in hierarchical sensor networks. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, volume 1, page 8, 2006. doi: 10.1109/SUTC.2006.1636206.

[72] J. Hill et al. System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[73] G. Hoblos, M. Staroswiecki, and A. Aitouche. Optimal design of fault tolerant sensor networks. In *Control Applications, 2000. Proceedings of the 2000 IEEE International Conference on*, pages 467–472, 2000. doi: 10.1109/CCA.2000.897468.

[74] Xiaoyan Hong, Pu Wang, Jiejun Kong, Qunwei Zheng, and jun Liu. Effective probabilistic approach protecting sensor traffic. In *Military Communications Conference, 2005. MILCOM 2005. IEEE*, volume 1, pages 169–175, 2005. doi: 10.1109/MILCOM.2005.1605681.

[75] Jun Hu, Zeng Ma, and Chunsheng Sun. Energy-efficient MAC protocol designed for wireless sensor network for iot. In *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, pages 721–725, December 2011. doi: 10.1109/CIS.2011.164.

[76] Jonathan W. Hui and David E. Culler. IP is dead, long live IP for wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 15–28, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-990-6. doi: 10.1145/1460412.1460415.

[77] Arshad Jhumka, Matthew Leeke, and Sambid Shrestha. On the use of fake sources for source location privacy: Trade-offs between energy and privacy. *The Computer Journal*, 54(6):860–874, 2011. doi: 10.1093/comjnl/bxr010.

[78] Arshad Jhumka, Matthew Bradbury, and Sain Saginbekov. Efficient fault-tolerant collision-free data aggregation scheduling for wireless sensor networks. *Journal of Parallel and Distributed Computing*, 74(1):1789 – 1801, 2014. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2013.09.011. URL `http://www.sciencedirect.com/science/article/pii/S0743731513002074`.

[79] Arshad Jhumka, Matthew Bradbury, and Matthew Leeke. Fake source-based source location privacy in wireless sensor networks. *Concurrency and Computation: Practice and Experience*, 27(12):2999–3020, 2015. ISSN 1532-0634. doi: 10.1002/cpe.3242.

[80] Pandurang Kamat, Yanyong. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 599–608, June 2005. doi: 10.1109/ICDCS.2005.31.

[81] Issa Khalil, Saurabh Bagchi, and Cristina Nina-Rotaru. Dicas: Detection, diagnosis and isolation of control attacks in sensor networks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 89–100, September 2005. doi: 10.1109/SECURECOMM.2005.17.

[82] J. Kirton, M. Bradbury, and A. Jhumka. Source location privacy-aware data aggregation scheduling for wireless sensor networks. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2200–2205, June 2017. doi: 10.1109/ICDCS.2017.171.

[83] Jack Kirton, Matthew Bradbury, and Arshad Jhumka. Towards optimal source location privacy-aware tdma schedules in wireless sensor networks. *Computer Networks*, 146:125 – 137, 2018. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2018.09.010. URL `http://www.sciencedirect.com/science/article/pii/S1389128618308958`.

[84] K. D. Lee and T. S. P. Yum. On pareto-efficiency between profit and utility in ofdm resource allocation. *IEEE Transactions on Communications*, 58(11): 3277–3285, November 2010. ISSN 0090-6778. doi: 10.1109/TCOMM.2010.091310.0900102.

[85] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In Werner Weber, JanM. Rabaey, and Emile Aarts, editors, *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-23867-6. doi: 10.1007/3-540-27139-2_7.

[86] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 126–137, New York, NY, USA, 2003. ACM. ISBN 1-58113-707-9. doi: 10.1145/958491.958506.

[87] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, pages 153–166, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1959-1. doi: 10.1145/2461381.2461402. URL `http://doi.acm.org/10.1145/2461381.2461402`.

[88] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 52–61, New York, NY, USA, 2003. ACM. ISBN 1-58113-738-9. doi: 10.1145/948109.948119.

[89] Jun Long, Mianxiong Dong, K. Ota, and Anfeng Liu. Achieving source location privacy and network lifetime maximization through tree-based diversionary routing in wireless sensor networks. *IEEE Access*, 2:633–651, 2014. ISSN 2169-3536. doi: 10.1109/ACCESS.2014.2332817.

[90] K. Mehta, Donggang Liu, and M. Wright. Location privacy in sensor networks against a global eavesdropper. In *2007 IEEE International Conference on Network Protocols*, pages 314–323, October 2007. doi: 10.1109/ICNP.2007.4375862.

[91] Hongxia Miao, Xuanxuan Xiao, Bensheng Qi, and Kang Wang. Improvement and application of leach protocol based on genetic algorithm for wsn. In *Computer Aided Modelling and Design of Communication Links and Networks (CAMAD), 2015 IEEE 20th International Workshop on*, pages 242–245, Sept 2015. doi: 10.1109/CAMAD.2015.7390517.

[92] P. Nagarathna and R. Manjula. Genetic algorithm with a new fitness function to enhance wsn lifetime. In *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pages 95–100, Oct 2015. doi: 10.1109/ICATCCT.2015.7456862.

[93] Xiaoguang Niu, Yihao Zhang, Yalan Yao, Xu Chen, Josep Miquel Jornet, and Jin Liu. An energy-efficient source-anonymity protocol in surveillance systems. *Personal and Ubiquitous Computing*, 20(5):771–783, 2016. ISSN 1617-4917. doi: 10.1007/s00779-016-0949-1.

[94] Fredrik Österlind. A sensor network simulator for the contiki os. Technical report, SICS publications database [http://eprints.sics.se/perl/oai2] (Sweden), 2006.

[95] Celal Ozturk, Yanyong Zhang, and Wade Trappe. Source-location privacy in energy-constrained sensor network routing. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, SASN '04, pages 88–93, New York, NY, USA, 2004. ACM. ISBN 1-58113-972-1. doi: 10.1145/1029102.1029117.

[96] Christian S. Perone. Pyevolve: A python open-source framework for genetic algorithms. *SIGEVOlution*, 4(1):12–20, November 2009. ISSN 1931-8499. doi: 10.1145/1656395.1656397.

[97] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: Security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, September 2002. ISSN 1022-0038. doi: 10.1023/A:1016598314198.

[98] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, June 2004. ISSN 0001-0782. doi: 10.1145/990680.990707.

[99] Ruben Rios, Javier Lopez, and Jorge Cuellar. *Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures*, chapter Location Privacy in WSNs: Solutions, Challenges, and Future Trends, pages 244–282. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10082-1. doi: 10.1007/978-3-319-10082-1_9.

[100] T. Sander and C. Tschudin. Towards mobile cryptography. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1998. IEEE Computer Society Press.

[101] Riaz Ahmed Shaikh, Hassan Jameel, Brian J. D'Auriol, Heejo Lee, Sungyoung Lee, and Young-Jae Song. Achieving network level privacy in wireless sensor networks. *Sensors*, 10(3):1447–1472, 2010. ISSN 1424-8220. doi: 10.3390/s100301447.

[102] Min Shao, Yi Yang, Sencun Zhu, and Guohong Cao. Towards statistically strong source anonymity for sensor networks. In *INFOCOM, 2008 Proceedings IEEE*, April 2008. doi: 10.1109/INFOCOM.2008.19.

[103] Min Shao, Wenhui Hu, Sencun Zhu, Guohong Cao, S. Krishnamurth, and T. La Porta. Cross-layer enhanced source location privacy in sensor networks. In *6^{th} Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '09)*, pages 1–9, 2009. doi: 10.1109/SAHCN.2009.5168923.

[104] Rui Shi, M. Goswami, Jie Gao, and Xianfeng Gu. Is random walk truly memoryless — traffic analysis and source location privacy under random walks. In *INFOCOM, 2013 Proceedings IEEE*, pages 3021–3029, April 2013. doi: 10.1109/INFCOM.2013.6567114.

[105] Sergei P. Skorobogatov. Semi-invasive attacks – A new approach to hardware security analysis. Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, April 2005. URL `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf`.

[106] R. Srivathsan, S. Siddharth, R. Muthuregunathan, R. Gunasekaran, and V. R. Uthariaraj. Enhanced genetic algorithm for solving broadcast scheduling problem in tdma based wireless networks. In *2010 Second International Conference on COMmunication Systems and NETworks (COMSNETS 2010)*, pages 1–10, Jan 2010. doi: 10.1109/COMSNETS.2010.5431986.

[107] Wei Tan, Ke Xu, and Dan Wang. An anti-tracking source-location privacy protection protocol in WSNs based on path extension. *Internet of Things Journal, IEEE*, 1(5):461–471, October 2014. ISSN 2327-4662. doi: 10.1109/JIOT.2014.2346813.

[108] Hui Tian, Hong Shen, and Matthew Roughan. Maximizing networking lifetime in wireless sensor networks with regular topologies. In *9th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 211–217. IEEE, 2008.

[109] R Valli and P Dananjayan. Utility enhancement by game theoretic approach using square grid topology in wsn. In *International Conference on Process Automation, Control and Computing*, pages 1–4. IEEE, 2011.

[110] J. Viega, J. T. Bloch, Y. Kohno, and Gary McGraw. Its4: a static vulnerability scanner for c and c++ code. In *Computer Security Applications, 2000. ACSAC '00. 16th Annual Conference*, pages 257–267, 2000. doi: 10.1109/ACSAC.2000.898880.

[111] Wang Wei-Ping, Chen Liang, and Wang Jian-xin. A source-location privacy protocol in WSN based on locational angle. In *IEEE International Conference on Communications (ICC)*, pages 1630–1634, May 2008. doi: 10.1109/ICC.2008.315.

[112] Yong Xi, L. Schwiebert, and Weisong Shi. Preserving source location privacy in monitoring-based wireless sensor networks. In *20th International Parallel and Distributed Processing Symposium*, pages 1–8, April 2006. doi: 10.1109/IPDPS.2006.1639682.

[113] Lin Yao, Lin Kang, Fangyu Deng, Jing Deng, and Guowei Wu. Protecting source–location privacy based on multirings in wireless sensor networks. *Con-*

currency and Computation: Practice and Experience, 27(15):3863–3876, 2015. ISSN 1532-0634. doi: 10.1002/cpe.3075.

[114] Liang Zhang. A self-adjusting directed random walk approach for enhancing source-location privacy in sensor network routing. In *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, IWCMC '06, pages 33–38, New York, NY, USA, 2006. ACM. ISBN 1-59593-306-9. doi: 10.1145/1143549.1143558.

[115] O. Zorlu and O. K. Sahingoz. Increasing the coverage of homogeneous wireless sensor network by genetic algorithm based deployment. In *2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pages 109–114, July 2016. doi: 10.1109/DICTAP.2016.7544010.